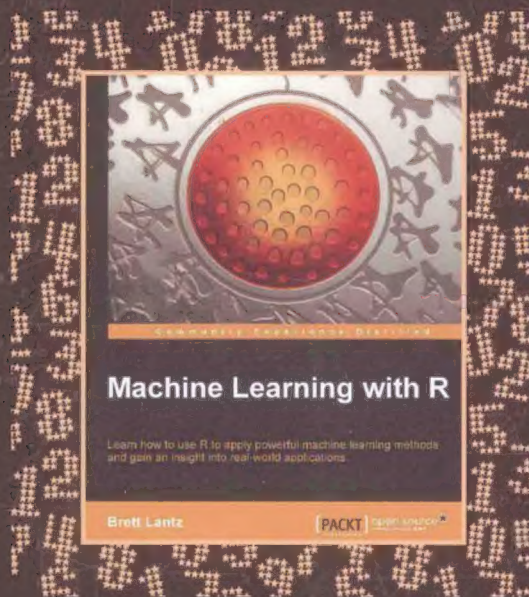


机器学习与R语言

[美] Brett Lantz 著

李洪成 许金炜 李舰 译



MACHINE LEARNING WITH R

机器学习与R语言

MACHINE LEARNING WITH R

随着大数据的概念变得越来越流行，对数据的探索、分析和预测成为大数据分析领域的基本技能之一。作为探索和分析数据的基本理论和工具，机器学习和数据挖掘成为时下炙手可热的技术。R作为功能强大并且免费的数据分析工具，在数据分析领域获得了越来越多用户的青睐。

本书通过丰富的实际案例来探索如何应用R来进行现实世界问题的机器学习，如何从数据中获取可以付诸行动的洞察力。本书案例清晰而实用，讲解循序渐进，是一本用R进行机器学习的实用指南，既适用于机器学习的初学者，也适用于具有一定经验的老手，本书将帮助他们回答有关R的所有问题。

通过阅读本书，你将学到：

- 用R准备用于机器学习的数据
- 用R进行数据探索和数据可视化
- 用k近邻方法进行数据分类
- 应用朴素贝叶斯方法进行数据分类
- 应用决策树、规则和支持向量机进行预测
- 用线性回归预测数值型数据
- 用神经网络对数据建模
- 应用购物篮分析的关联规则找出数据中的模式
- 通过对数据聚类进行市场细分

[PACKT]
PUBLISHING



投稿热线: (010) 88379604
客服热线: (010) 88378991 88361066
购书热线: (010) 68326294 88379649 68995259

华章网站: www.hzbook.com
网上购书: www.china-pub.com
数字阅读: www.hzmedia.com.cn

上架指导: 计算机/机器学习

ISBN 978-7-111-49157-6



9 787111 491576 >

定价: 69.00元

数据科学

MACHINE LEARNING WITH R
机器学习与R语言

[美] Brett Lantz 著

李洪成 许金炜 李舰 译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

机器学习与 R 语言 / (美) 兰兹 (Lantz, B.) 著; 李洪成, 许金炜, 李舰译. — 北京: 机械工业出版社, 2015.1

(数据科学与工程丛书)

书名原文: Machine Learning with R

ISBN 978-7-111-49157-6

I. 机… II. ①兰… ②李… ③许… ④李… III. ①机器学习 ②程序语言—程序设计
IV. ① TP181 ② TP312

中国版本图书馆 CIP 数据核字 (2015) 第 013891 号

本书版权登记号: 图字: 01-2013-9378

Brett Lantz: Machine Learning with R (ISBN: 978-1-78216-214-8)

Copyright © 2013 Packt Publishing. First published in the English language under the title "BackTrack 4: Assuring Security by Penetration Testing".

All rights reserved.

Chinese simplified language edition published by China Machine Press.

Copyright © 2015 by China Machine Press.

本书中文简体字版由 Packt Publishing 授权机械工业出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

机器学习与 R 语言

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 盛思源

责任校对: 董纪丽

印 刷: 三河市宏图印务有限公司

版 次: 2015 年 3 月第 1 版第 1 次印刷

开 本: 186mm × 240mm 1/16

印 张: 17.25

书 号: ISBN 978-7-111-49157-6

定 价: 69.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzjsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

You have probably encountered the output of machine learning in many ways. When you read your email, spam has probably already been filtered out by a machine learning algorithm - most likely a Bayesian one. When you surf the web, you will see ads generated by machine learning models that predict what is likely to appeal to you. When you apply for a loan or credit card, approval will depend on the output of a machine learning model. When someone steals your credit card, the bank is hoping that its machine learning algorithm will identify fraudulent purchases as soon as they happen.

In its early days, the predictive modeling of machine learning was the province of very expensive statistical software, often bundled with consulting services to set up data mining systems. The advent of R has brought machine learning within the reach of smaller companies, startups and even individuals. Many big data-oriented companies now rely primarily on open source tools like R to deploy machine learning.

This book can be your guide to learning about the business context of machine learning, and also to the actual implementation of machine learning methods using R.

[你可能以多种方式接触过机器学习的输出结果。当你阅读电子邮件时，垃圾邮件可能已经被某个机器学习算法（很可能是贝叶斯算法）过滤掉了；当你在网页上浏览时，你可能会看到一些广告，它们是由机器学习算法预测出的可能会吸引你的广告；当你申请贷款或者信用卡时，申请的批准与否取决于机器学习模型的输出结果；当有人盗取了你的信用卡，发卡银行希望当该卡被用于欺诈消费时他们部署的机器学习算法能够识别出该类偷盗消费。

在早些时候，具有机器学习的预测模型只有很昂贵的统计软件才涉及，它们经常和咨询服务一起作为数据挖掘系统的一部分。R 软件出现之后，小公司、初创公司，甚至个人都开

始应用机器学习。现在，很多面向数据的大公司主要依靠像 R 软件这样的开源工具来部署他们的机器学习应用。

本书既可以作为你了解机器学习应用的商业背景的指南，也可以作为应用 R 来实现机器学习方法的指导。]



美国统计教育学院, Statistics.com 在线课程网站总裁

随着大数据的概念变得越来越流行，对数据的探索、分析和预测成为大数据分析领域的基本技能之一。作为探索和分析数据的基本理论和工具，机器学习和数据挖掘成为时下的热门技术之一。R 作为功能强大并且免费的数据分析工具，在数据分析领域获得了越来越多用户的青睐。本书介绍如何应用 R 来进行现实世界问题的机器学习，以及如何从数据中获取可以付诸行动的洞察力。

本书的作者 Brett Lantz 在机器学习领域具有十余年的实践经验。他在本书中介绍了多种重要的机器学习算法。在给出相应的机器学习算法的核心理论之后，都给出了一个实际的案例，从对案例数据的探索、整理，到模型的建立和模型的评估，每一步都给出了详尽的步骤和 R 代码。

本书共分 12 章。第 1 章介绍机器学习的基本概念和理论，并介绍用于机器学习的 R 软件环境的准备。第 2 章介绍如何应用 R 来管理数据，进行数据的探索分析和数据可视化。第 3 章到第 9 章介绍典型的机器学习算法和案例，包括：k 近邻分类算法、朴素贝叶斯算法、决策树和规则树、回归预测、黑盒算法——神经网络和支持向量机、关联分析、k 均值聚类。伴随着这些算法的介绍，书中给出了大量的实际案例，并给出了详细的分析步骤，例如乳腺癌的判断、垃圾短信的过滤、贷款违约的预测、毒蘑菇的判别、医疗费用的预测、建筑用混凝土强度的预测、光学字符的识别、超市购物篮关联分析以及市场细分等。第 10 章介绍模型性能评价的原理和方法。第 11 章给出提高模型性能的几种常用方法。第 12 章讨论用 R 进行机器学习时可能遇到的一些高级专题，例如特殊形式的数据、大数据集的处理、并行计算和 CPU 计算等技术。

R 本身是一款十分优秀的数据分析和数据可视化软件，其中包括大量用于机器学习的添加包。本书以机器学习算法为主线，通过案例学习的形式来组织内容，脉络清晰，并且各章自成体系。读者可以从头逐章学习，也可以找到自己所需要的内容进行学习。读者只需要具有 R 的一些基本知识，不需要具备机器学习的深厚基础。不管是 R 初学者，还是熟练的 R

用户都能从书中找到对自己有用的内容。

译者曾经应用本书的部分内容进行教学，学生都反映这些内容具有极强的实用价值，许多内容可以直接或者略加修改就可以应用到他们的实际工作中。我们有幸受机械工业出版社委托将此书译成中文，希望中文版的出版能够给国内读者学习 R 与机器学习带来方便。

在本书的翻译过程中，得到了王春华编辑的大力支持和帮助。本书责任编辑盛思源老师具有丰富的经验，为本书的出版付出了大量的劳动，这里对她们的支持和帮助表示衷心的感谢。本书的翻译工作由李洪成、许金炜和李舰共同完成，丁一飞协助翻译了本书的部分内容，全书由李洪成进行修改并统一定稿。

由于时间和水平所限，难免会有不当之处，希望同行和读者多加指正。

李洪成

机器学习的核心是将信息转化为可行动智能的算法。这一事实使得机器学习非常适合于当今的大数据时代。如果没有机器学习，要跟上海量信息数据流的步伐几乎是不可能的。

鉴于 R 不断增长的地位（R 是一个跨平台、零成本的统计编程环境），现在是开始使用机器学习的最好时代。R 提供了一套功能强大且易于学习的工具，这些工具可以帮助你发现数据背后隐藏的信息。

本书通过将实际案例与核心理论知识相结合，提供了你开始将机器学习应用到你自己的项目中所需要的知识。

本书内容

第 1 章介绍了用来定义和区分机器学习算法的术语和概念，并给出将学习任务与适当算法相匹配的方法。

第 2 章提供了一个在 R 中自己实际动手操作数据的机会，并讨论了基本的数据结构以及用于加载、探索和理解数据的程序。

第 3 章教你如何理解并将一个简单且功能强大的学习算法应用于你的第一个学习任务：识别乳腺癌。

第 4 章揭示了用于先进的垃圾邮件过滤系统中的概率的基本概念，并且在你自己建立垃圾邮件过滤器的过程中，你将学习文本挖掘的基本知识。

第 5 章探索几种预测精度高且容易解释的学习算法。我们将把这几种算法应用于对透明度要求很高的学习任务中。

第 6 章介绍用于数值预测的机器学习算法。由于这些技术在很大程度上来源于统计学领域，所以你还学习理解数值之间关系的必要分析指标。

第 7 章介绍两个极其复杂但功能强大的机器学习算法。尽管其中的数学理论可能会让人望而生畏，但是我们将以简单的术语，通过例子来说明这些方法的内部运作原理。

第 8 章揭示许多零售商所使用的推荐系统的算法。如果你曾经想知道零售商如何比你自己更了解你的购物习惯，该章将揭示他们的秘密。

第 9 章介绍 k 均值聚类，该算法用于找出相关个体的聚类。我们将使用该算法来确定一个基于网络的社区特征的分区。

第 10 章提供度量机器学习项目是否成功的信息，并给出了机器学习算法在未来数据上性能的一个可靠的估计。

第 11 章揭示了在机器学习竞赛中排名最靠前的团队所采用的方法。如果你具有竞争意识，或者仅仅想获取数据中尽可能多的信息，那么你需要将这些技术添加到你的知识库中。

第 12 章讨论机器学习的前沿主题。从使用大数据到如何使 R 运行速度更快，这些主题将会帮助你拓展使用 R 进行数据挖掘的界限。

学习本书的准备知识

本书的例子是基于微软的 Windows 系统和 Mac OS X 系统的 R 2.15.3 进行编写和测试的，不过对于任意最新的 R 版本，这些例子基本上都能运行。

本书适用对象

本书适合于任何希望使用数据来采取行动的人。或许你已经对机器学习有些了解，但从来没有使用过 R；或许你已经对于 R 有些了解，但机器学习对你来说是全新的知识。无论何种情况，本书将让你快速上手。稍微了解一些基本的数学知识和编程概念将是有帮助的，但是这些先验知识并不是必需的，你需要具有的就是好奇心。

Acknowledgements 致谢

没有我家人和朋友的支持，写作本书是根本不可能的。尤其是，非常感谢我的妻子 Jessica 在过去的一年中对我的耐心与鼓励。感谢我的儿子 Will（他出生时我正在写作第 10 章），特别值得一提的是，他在我写作过程中所扮演的角色，如果没有他一觉睡到天明的恩典，我就是到了第二天早上也根本不可能写出一个连贯的句子。我把本书献给他，希望有一天，他的灵感能跟随好奇心的驱动，无论好奇心会指向哪里。

我还要感谢支持本书的很多人。书中要表达的很多想法来源于我与 University of Michigan（密歇根大学）、University of Notre Dame（圣母玛利亚大学）以及 University of Central Florida（佛罗里达中央大学）的教育工作者、同事以及合作者的交流。此外，如果没有各位学者以公开出版物、课程和代码的方式分享他们的知识，本书可能根本就不会存在。最后，我感谢 R 团队和所有那些贡献 R 添加包的人的努力，是他们的努力最终为大家带来了机器学习。

关于技术评审人 *About the Reviewers*

Jia Liu 拥有巴尔的摩马里兰大学 (the University of Maryland, Baltimore County) 统计学硕士学位, 她目前是爱荷华州立大学 (Iowa State University) 的统计学博士候选人。她的研究领域包括混合效应模型 (mixed-effects model)、贝叶斯方法 (Bayesian method)、自助法 (Bootstrap method)、可靠性 (reliability)、试验设计 (design of experiments)、机器学习和数据挖掘。她有两年的学生统计学顾问的经验, 并且在农业和制药业有两年的实习经验。

Mzabalazo Z. Ngwenya 广泛地工作在统计咨询领域, 他目前是一名生物统计学家。他拥有开普敦大学 (University of Cape Town) 数理统计理学硕士学位, 目前正在攻读计算智能领域的博士学位 (在南非比勒陀利亚大学信息技术学院)。他的研究领域包括统计计算 (statistical computing)、机器学习和空间统计学 (spatial statistics)。此前, 他曾参与审阅下面的著作: 《Learning RStudio for R Statistical Computing》(Van de Loo 和 de Jong, 2012) 和 《R Statistical Application Development by Example beginner's guide》(Prabhanjan Narayanachar Tattar, 2013)。

Abhinav Upadhyay 在 2011 年获得了信息技术专业的学士学位, 他感兴趣的领域主要有机器学习和信息检索。

2011 年, 他曾为 NetBSD Foundation (谷歌公司夏季代码 (Google Summer of Code) 方案一部分) 工作。在此期间, 他为 UNIX 手册页编写了一个搜索引擎, 该项目为 NetBSD 的 apropos 功能给出了一个新的实现方法。

目前, 他是 SocialTwist 公司的一名开发工程师。他的日常工作涉及编写系统工具与框架来管理产品的基础架构。

他也是一个开源爱好者, 在社区相当活跃。在业余时间, 他维护和效力于多个开源项目。

推荐阅读



数据挖掘：概念与技术（第3版）

作者：Jiawei Han 等 译者：范明 等 ISBN：978-7-111-39140-1 定价：79.00元

英文版：978-7-111-37431-2 定价：118.00元

数据挖掘：实用机器学习工具与技术（原书第3版）

作者：Ian H. Witten 等 译者：李川 等 ISBN：978-7-111-45381-9 定价：79.00元

数据挖掘导论（英文版）

作者：Pang-Ning Tan 等 ISBN 978-7-111-31670-1 定价 59.00元

社交网站的数据挖掘与分析

作者：Matthew A. Russell ISBN 978-7-111-36960-8 定价 59.00元

机器学习

作者 Tom Mitchell ISBN 978-7-111-10993-7 定价 35.00元
英文版：7-111-11502-3 定价：58.00元

数据挖掘：实用机器学习工具与技术（英文版·第3版）

作者 Ian H. Witten 等 ISBN：978-7-111-37417-6 定价 108.00元

推荐阅读



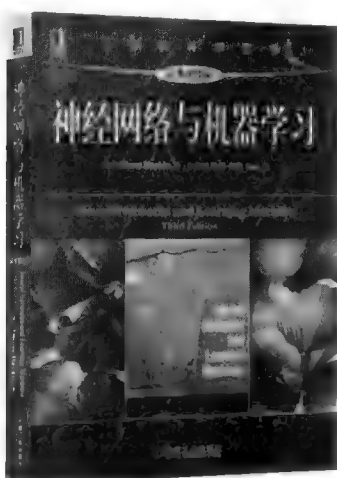
机器学习

作者 (美) Tom Mitchell ISBN 978-7-111-10993-7 定价: 35.00元



机器学习基础教程

作者 (英) Simon Rogers 等 ISBN: 978-7-111-40702-7 定价 45.00元



神经网络与机器学习 (原书第3版)

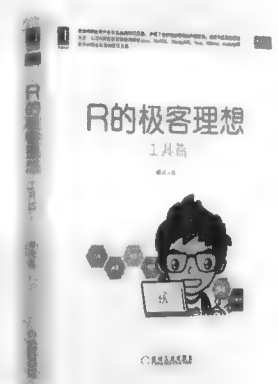
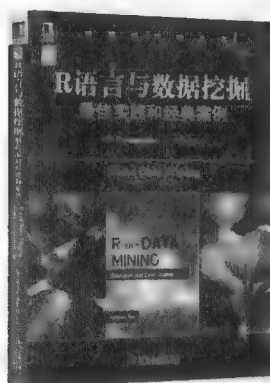
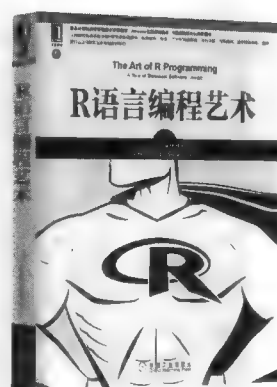
作者: (加) Simon Haykin ISBN: 978-7-111-32413-3 定价: 79.00元



模式分类 (原书第2版)

作者 (美) Richard O. Duda 等 ISBN: 978-7-111-12148-1 定价: 59.00元

推荐阅读



数据挖掘与R语言

作者: Luis Torgo ISBN: 978-7-111-40700-3 定价: 49.00元

R语言经典实例

作者: Paul Teetor ISBN: 978-7-111-42021-7 定价: 79.00元

R语言编程艺术

作者: Norman Matloff ISBN: 978-7-111-42314-0 定价: 69.00元

R语言与数据挖掘最佳实践和经典案例

作者: Yanchang Zhao ISBN: 978-7-111-47541-5 定价: 49.00元

R语言与网站分析

作者: 李明 ISBN: 978-7-111-45971-2 定价: 79.00元

R的极客理想——工具篇

作者: 张丹 ISBN: 978-7-111-47507-1 定价: 59.00元

Contents 目 录

推荐序	2.2 向量	19
译者序	2.3 因子	20
前言	2.3.1 列表	21
致谢	2.3.2 数据框	22
关于技术评审人	2.3.3 矩阵和数组	24
第1章 机器学习简介	2.4 用 R 管理数据	25
1.1 机器学习的起源	2.4.1 保存和加载 R 数据结构	25
1.2 机器学习的使用与滥用	2.4.2 用 CSV 文件导入和保存数据	26
1.3 机器如何学习	2.4.3 从 SQL 数据库导入数据	27
1.3.1 抽象化和知识表达	2.5 探索和理解数据	28
1.3.2 一般化	2.5.1 探索数据的结构	29
1.3.3 评估学习的成功性	2.5.2 探索数值型变量	29
1.4 将机器学习应用于数据中的步骤	2.5.3 探索分类变量	37
1.5 选择机器学习算法	2.5.4 探索变量之间的关系	39
1.5.1 考虑输入的数据	2.6 总结	42
1.5.2 考虑机器学习算法的类型		
1.5.3 为数据匹配合适的算法		
1.6 使用 R 进行机器学习		
1.7 总结		
第2章 数据的管理和理解	第3章 懒惰学习——使用近邻分类	44
2.1 R 数据结构	3.1 理解使用近邻进行分类	45
	3.1.1 kNN 算法	45
	3.1.2 为什么 kNN 算法是懒惰的	51
	3.2 用 kNN 算法诊断乳腺癌	51
	3.2.1 第1步——收集数据	51

3.2.2	第2步——探索和准备数据	52
3.2.3	第3步——基于数据训练模型	55
3.2.4	第4步——评估模型的性能	57
3.2.5	第5步——提高模型的性能	58
3.3	总结	60

第4章 概率学习——朴素贝叶斯分类

4.1	理解朴素贝叶斯	61
4.1.1	贝叶斯方法的基本概念	62
4.1.2	朴素贝叶斯算法	65
4.2	例子——基于贝叶斯算法的手机垃圾短信过滤	70
4.2.1	第1步——收集数据	70
4.2.2	第2步——探索和准备数据	71
4.2.3	数据准备——处理和分析文本数据	72
4.2.4	第3步——基于数据训练模型	78
4.2.5	第4步——评估模型的性能	79
4.2.6	第5步——提升模型的性能	80
4.3	总结	81

第5章 分而治之——应用决策树和规则进行分类

5.1	理解决策树	82
5.1.1	分而治之	83
5.1.2	C5.0 决策树算法	86
5.2	例子——使用 C5.0 决策树识别高风险银行贷款	89
5.2.1	第1步——收集数据	89

5.2.2	第2步——探索和准备数据	89
5.2.3	第3步——基于数据训练模型	92
5.2.4	第4步——评估模型的性能	95
5.2.5	第5步——提高模型的性能	95

5.3 理解分类规则

5.3.1	独立而治之	99
5.3.2	单规则 (1R) 算法	101
5.3.3	RIPPER 算法	103
5.3.4	来自决策树的规则	105

5.4 例子——应用规则学习识别有毒的蘑菇

5.4.1	第1步——收集数据	106
5.4.2	第2步——探索和准备数据	106
5.4.3	第3步——基于数据训练模型	107
5.4.4	第4步——评估模型的性能	109
5.4.5	第5步——提高模型的性能	109

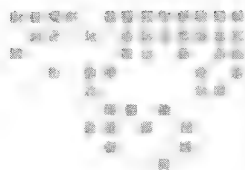
5.5 总结

第6章 预测数值型数据——回归方法

6.1	理解回归	113
6.1.1	简单线性回归	115
6.1.2	普通最小二乘估计	117
6.1.3	相关系数	118
6.1.4	多元线性回归	120

6.2 例子——应用线性回归预测医疗费用.....122	7.2.4 第4步——评估模型的性能.....158
6.2.1 第1步——收集数据.....122	7.2.5 第5步——提高模型的性能.....159
6.2.2 第2步——探索和准备数据.....123	7.3 理解支持向量机.....160
6.2.3 第3步——基于数据训练模型.....127	7.3.1 用超平面分类.....161
6.2.4 第4步——评估模型的性能.....129	7.3.2 寻找最大间隔.....161
6.2.5 第5步——提高模型的性能.....130	7.3.3 对非线性空间使用核函数.....164
6.3 理解回归树和模型树.....133	7.4 用支持向量机进行光学字符识别.....165
6.4 例子——用回归树和模型树估计葡萄酒的质量.....135	7.4.1 第1步——收集数据.....166
6.4.1 第1步——收集数据.....135	7.4.2 第2步——探索和准备数据.....166
6.4.2 第2步——探索和准备数据.....136	7.4.3 第3步——基于数据训练模型.....167
6.4.3 第3步——基于数据训练模型.....137	7.4.4 第4步——评估模型的性能.....169
6.4.4 第4步——评估模型的性能.....140	7.4.5 第5步——提高模型的性能.....170
6.4.5 第5步——提高模型的性能.....142	7.5 总结.....171
6.5 总结.....144	
第7章 黑箱方法——神经网络和支持向量机.....146	
7.1 理解神经网络.....146	第8章 探寻模式——基于关联规则的购物篮分析.....172
7.1.1 从生物神经元到人工神经元.....148	8.1 理解关联规则.....172
7.1.2 激活函数.....148	8.2 例子——用关联规则确定经常一起购买的食物杂货.....176
7.1.3 网络拓扑.....151	8.2.1 第1步——收集数据.....176
7.1.4 用后向传播训练神经网络.....153	8.2.2 第2步——探索和准备数据.....177
7.2 用人工神经网络对混凝土的强度进行建模.....154	8.2.3 第3步——基于数据训练模型.....183
7.2.1 第1步——收集数据.....154	
7.2.2 第2步——探索和准备数据.....155	
7.2.3 第3步——基于数据训练模型.....156	

8.2.4 第4步——评估模型的性能·····	184	10.3 总结·····	229
8.2.5 第5步——提高模型的性能·····	187		
8.3 总结·····	189		
第9章 寻找数据的分组——k 均值		第11章 提高模型的性能 ·····	231
聚类 ·····	191	11.1 调整多个模型来提高性能·····	231
1.1 理解聚类·····	191	11.2 使用元学习来提高模型的	
1.1.1 聚类——一种机器学习任务·····	192	性能·····	239
1.1.2 k 均值聚类算法·····	193	11.2.1 理解集成学习·····	239
1.1.3 用 k 均值聚类探寻青少年市场		11.2.2 bagging·····	241
细分·····	198	11.2.3 boosting·····	243
1.1.4 第1步——收集数据·····	198	11.2.4 随机森林·····	244
1.1.5 第2步——探索和准备数据·····	199	11.3 总结·····	248
1.1.6 第3步——基于数据训练			
模型·····	202	第12章 其他机器学习主题 ·····	249
1.1.7 第4步——评估模型的性能·····	204	12.1 分析专用数据·····	250
1.1.8 第5步——提高模型的性能·····	206	12.1.1 用 R Curl 添加包从网上	
9.2 总结·····	207	获取数据·····	250
		12.1.2 用 XML 添加包读 / 写 XML	
		格式数据·····	250
		12.1.3 用 rjson 添加包读 / 写	
		JSON·····	251
第10章 模型性能的评价 ·····	208	12.1.4 用 xlsx 添加包读 / 写	
10.1 度量分类方法的性能·····	208	Microsoft Excel 电子表格·····	251
10.1.1 在 R 中处理分类预测数据·····	209	12.1.5 生物信息学数据·····	251
10.1.2 深入探讨混淆矩阵·····	211	12.1.6 社交网络数据和图数据·····	252
10.1.3 使用混淆矩阵度量性能·····	212	12.2 提高 R 语言的性能·····	252
10.1.4 准确度之外的其他性能评价		12.2.1 处理非常大的数据集·····	253
指标·····	214	12.2.2 使用并行处理来加快	
10.1.5 性能权衡的可视化·····	221	学习过程·····	254
10.2 评估未来的性能·····	224	12.2.3 GPU 计算·····	257
10.2.1 保持法·····	225	12.2.4 部署最优的学习算法·····	257
10.2.2 交叉验证·····	226	12.3 总结·····	258
10.2.3 自助法抽样·····	229		



机器学习简介

如果科幻故事是可信的，那么教会机器学习将会不可避免地导致机器和其制造者之间的末日战争。在计算机使用的早期，计算机被教会玩井字棋和国际象棋这样一些简单的游戏。后来，机器被用来控制交通信号灯和通信，随后用来控制军用无人机和导弹。一旦计算机有感知力并且知道如何教会他们自己，机器的发展将产生不祥的改变：计算机不再需要人类程序员了，人类那时也就被“删除”(deleted)了。

幸运的是，在写本书的时候，机器还是需要用户来进行输入的。

人们对机器学习的印象可能深受那些大众媒体中所描述的人工智能故事的影响。尽管这些故事中有真实世界的迹象，但实际上机器学习注重于更加现实的应用。教会计算机学习这项工作更多时候是与特定问题联系在一起，比如说让计算机学会玩游戏、探究哲学或者回答一些简单的问题。机器学习更像是培训员工，而不是养育孩子。

把这些套话放在一边，在本章结束时，你会对机器学习有更加清晰的理解。本章将介绍一些基本概念，通过它们来定义和区分常用的机器学习方法。

本章中，你将会学到下列知识：

- ❑ 机器学习的起源及其实际应用。
- ❑ 知识是如何在计算机中定义和表达的。
- ❑ 用来区分机器学习方法的基本概念。

总之，可以说机器学习提供了应用计算机并把数据转换成可行动的知识工具集合。本书就是让你了解这一切是怎么实现的。

1.1 机器学习的起源

自从我们出生以来，我们就和各种数据打交道。我们身体的感官——眼睛、耳朵、鼻子、舌头以及神经一直被数据包围着，大脑把它们转化成视觉、听觉、嗅觉、味觉和感知。通过语言的交流，我们得以与他人分享这些感受。

最早的数据库记录了可观测环境中的信息。宇航员记录行星和恒星的模式；生物学家记录杂交植物和动物的实验结果；城市记录税收、疾病暴发情况以及人口数量。所有这些都要求人们先观察，然后再记录观察的结果。现在，由于不断发展的计算机数据库的应用，这些观察的过程逐步自动化，记录也变得系统化。

电子传感器的发明使得可以记录的数据更加丰富。专用的传感器可以观测、可以听声音、可以闻味道，也可以感受环境。这些传感器处理数据的方式和人类完全不同，很多情况下，这是传感器的优势。由于不用把原始数据转换成人类的语言，所以原始的传感器数据保留了数据的客观性。



很重要的一点是，尽管传感器不会对观测对象产生主观成分，但是它们也不一定报告真实情况（如果真实情况可以被界定的话）。与拍摄彩色照片的相机相比，拍摄黑白照片的相机可能会给出与拍摄环境完全不同的写照。类似地，显微镜对事实的描绘和望远镜的描绘也是截然不同的。

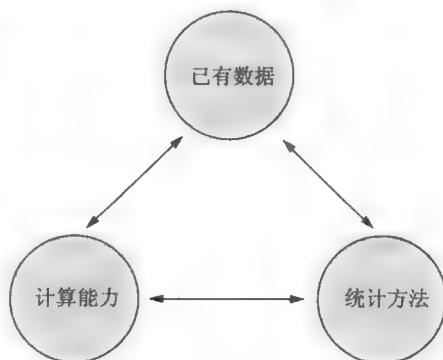
在数据库和传感器之间，我们生活的方方面面都被记录下来。记录政府、商业和个人信息并报告他们生活中各方面的信息。气象传感器记录温度和气压，监视探头监视着人行道和地铁站，各种电子行为如交易、通话、建立友好关系以及很多其他的行为都会被监控。

根据如此庞大的数据量，一些人声称我们进入了大数据的时代，这可能有一点哗众取宠。人类一直身处数据之中。使当今这个时代变得与众不同的是我们更容易获得数据。仅仅一次网络搜索，经过手指的点击，大量有趣的数据变得更容易获取。我们现在生活在一个大量数据可以直接被机器处理的时代。只要能够理解数据中存在的有规律的模式，这其中的很多信息都会成为有潜力的决策信息。

机器学习的研究领域是发明计算机算法，把数据转化为智能行为。这个领域是在现有数据、统计方法以及计算能力迅速并且同步发展的环境下发展起来的。数据量的增加使得计算能力增强成为必须条件，而计算能力的增强又反过来促进了分析大数据的统计方法的发展。这就创造了一个闭环式的发展，它使得更多更加有趣的数据得以收集。

机器学习的一个紧密相关的学科是数据挖掘，它涉及从大型的数据库中产生新的洞察（不要把它和

“挖掘数据”相混淆，“挖掘数据”是指挑选最合适的数据来支持某个理论的行为）。尽管对



于这两个领域究竟有多少重叠存在一些争议，但是一个可能的差别是机器学习侧重执行一个已知的任务，而数据挖掘则是侧重寻找有价值的信息。例如，你可能会用机器学习方法去教一个机器人开车，然而你会利用数据挖掘了解哪种类型的车是最安全的。



机器学习算法实际上是数据挖掘的先期准备，反之则不然。换句话说，你可以应用机器学习完成不涉及数据挖掘的任务，但是如果你应用数据挖掘方法，几乎肯定要用到机器学习。

1.2 机器学习的使用与滥用

机器学习的核心主要侧重于找出复杂数据的意义。这是一个应用广泛的任务，还有很多的应用方向没有被发现。就像你可能预期的那样，机器学习的使用是非常广泛的。例如，它已经用于：

- ☐ 预测选举的结果。
- ☐ 识别并且过滤垃圾邮件。
- ☐ 预测犯罪活动。
- ☐ 根据路况，实现交通信号灯的自动化
- ☐ 给出暴风雨和自然灾害后经济损失的估计。
- ☐ 检测客户流失。
- ☐ 设计自动驾驶飞机和自动驾驶汽车。
- ☐ 确定每个人捐助的能力。
- ☐ 把广告定位到特定类型的顾客。

现在还不用关心机器如何学习去执行这些任务，后面会具体介绍。上面提到的每种任务的背景不同，但是它们的机器学习过程是一样的。机器学习算法应用数据，找出那些可以应用到实际行动中的模式。在一些案例中，机器学习的结果非常成功，近乎达到了神奇的程度。

有一个可能是杜撰的故事，它讲到美国一家大型零售商使用机器学习来确定怀孕的母亲，并给她们发送有针对性的优惠券邮件。如果给准妈妈大量有针对性的折扣，零售商希望她们成为忠实的顾客，接下来将继续购买利润颇丰的商品，例如尿布、奶粉和玩具等。

通过应用机器学习方法分析客户购买数据，零售商相信他们可以得到一些有用的模式。可以通过一些商品，比如怀孕期的维生素、化妆水和毛巾等，以很大的确定性来判断一个女人是不是怀孕了，以及婴儿的产期是什么时候。

这家零售商根据这些数据发出促销邮件以后，一个生气的男人联系了他们，询问为什么他那只有十几岁的女儿收到了适用于孕妇的商品优惠券。他很生气，认为商人是在鼓励青少年怀孕。之后，一个经理出来道歉。然而最终道歉的却是这个父亲。他和他的女儿对质之后，发现她确实怀孕了。

无论前面的故事是否完全属实，这里有一点是真实的。零售商事实上会对他们的顾客交易记录进行例行分析。如果你曾经在超市、咖啡店或者其他商家那里使用过购物积分卡，那么你的购买记录数据很有可能已经被用作机器学习了。

零售商用机器学习方法来做广告，针对目标群体进行定向促销，进行存货管理或者规划商店里商品的布局。一些零售商甚至给结账通道配备了特别装置，它们可以根据本次交易购买的商品种类打印促销优惠券。网站也会基于你的网站浏览记录定期做类似的机器学习，据此进行广告发布。由于数据是从很多个体那里得到的，所以机器学习算法可以分析人们典型的行为模式，然后据此来给出商品推荐。

尽管你熟悉幕后的机器学习算法，但当零售商或者网站好像比我们自己还了解我们的时候，你还是感觉有些像魔术一样。有些人可能不太高兴以这种方式利用他们的信息。因此，任何人想要利用机器学习或者数据挖掘，至少应该考虑伦理道德方面的问题。

伦理方面的考虑

由于机器学习是一个相对较新的、正处于发展之中的学科，所以与之相对应的法律法规和社会准则尚不确定，并且一直在变化。获取和分析数据时要小心谨慎，避免违法、违反服务条例或者数据使用协议，避免滥用人们的信任，避免侵犯消费者或公众的隐私。



Google 可能是一个收集私人信息比其他任何同行都多的机构，其非正式的公司箴言是：不作恶。这可以作为你的一个比较合理的道德指引起点，但这也可能还不够。

某些司法规定可能不允许你把种族、民族、宗教或者其他一些受保护类型的数据用作商业用途，但是，你要牢记的是把这些数据排除在分析之外可能还不够，因为机器学习算法可能会不可避免地自己来学习这些信息。例如，如果某一特定群体的人一般生活在一个特定地区，购买特定的商品或者他们的行为能唯一地确定他们这个群体，那么一些机器学习算法可以从他们的一些表面上看无关紧要的数据中推算出那些受保护的信息。在这种情况下，除了受保护的信息之外，也要排除所有可能的潜在识别这群人的数据。

除了合法性的结果外，使用数据不当有可能会触及道德底线。如果一部分顾客认为是隐私的个人生活被公开的时候，他们可能会感到不舒服或者害怕。最近，当用户感到应用程序的服务协议条款发生变动，或者认为他们的数据被用于超出了他们原本同意应用程序可以使用的范围时，有几个备受瞩目的网络应用出现了大量的用户流失现象。对隐私的界定会由于内容、年龄层、地点的改变而不同，这就增加了合理使用个人数据的复杂性。在你开始你的项目之前先考虑文化差异的因素是很明智的。



事实上，你能够把数据应用于一个特定的目的并不总是代表你应该这样做。

1.3 机器如何学习

一个经常被引用的机器学习的正式定义是由计算机科学家 Tom M. Mitchell 提出的：如果机器能够获取经验并且能利用它们，在以后的类似经验中能够提高它的表现，该机器就称为机器学习。这个定义是相当精确的，其中提及机器学习技巧，以及实际中如何学习把数据转换成可行动的知识。



尽管在使用机器学习之前并不一定要先理解它的理论基础，但是这一基础能使人了解不同机器学习算法之间的区别。因为在人的大脑里，机器学习算法有许多不同的建模方式，你也可能通过从不同角度检验自己的思想来探索你自己。

无论学习者是人还是一个机器，基础的学习过程是类似的。这一过程可以分成如下 3 个部分：

- **数据输入**：这部分利用观察、记忆存储，以及回忆来提供进一步推理的事实依据。
- **抽象化**：这部分涉及把数据转换成更宽泛的表现形式（broader representation）。
- **一般化**：这部分应用抽象的数据来形成行动的基础。



为了更好地理解学习的过程，想象你最近参加的一场高难度的考试，该考试可能是大学里的一场期末考试或者一场资格证书考试。在考试之前，你想要过目不忘的本事吗（就像拍照那样）？如果你这么想过，当你知道过目不忘的本领不可能帮助你太多时，你可能会失望。如果没有深刻理解，你的知识就会局限于输入的数据。这就意味着你除了知道你见到过的之外，没有获取其他多余的信息。因此，如果不知道所有可能被测验问题的知识，你将局限于试图把所有会考的问题的答案都记下来。显然这种方式不具备持久性。

代替它的一种更好的方法是花时间有选择性地理解一小部分的核心概念。建立大纲或者概念图是通常应用的学习策略，这和机器执行知识抽象的方式是相类似的。这种方法界定了信息间的关系，并且这种方法描述了艰深的主题而不用逐字去记忆。这是一种更高级的学习方式，它要求学习者用他们自己的话来描述这些主题。

公布考试分数时总是一个紧张的时刻，学习策略也会被显然或者不显然地给出或高或低的分数。这里，你就会发现你的学习策略是否会扩展到老师或者教授考察的问题。一般化一方面需要大量的抽象数据，另一方面它也需要高度理解如何把这些知识应用到没有预知的考试题目中。一个优秀的教师在这方面会很有帮助。

记住，尽管我们把学习的过程阐述为 3 个不同的部分，但是这仅仅是为了便于阐述的目的才这么归类的。事实上，学习的这 3 个组成部分是紧密相连的。尤其是抽象化和一般化这

两个步骤的关系是如此紧密，以至于不可能在没有另一个步骤的情况下单独执行其中的一个步骤。对人类来说，整个过程都是下意识发生的。我们收集数据，推理数据，归纳数据，最后发现规律。然而对于一台计算机来说，这些步骤一定要分步进行。从另一个方面来说，这也是机器学习的一个优点。因为这个过程是透明化的，学到的知识能在以后的应用中得到检验和利用。

1.3.1 抽象化和知识表达

把原始数据用一个结构化的格式来表达是学习算法最典型的任务。在此之前，数据仅仅是硬盘上或者内存中的1和0（二进制），它们没有意义。给数据赋予含义的工作是在抽象化步骤中进行的。

观点（idea）和现实（reality）之间的联系可以用René Magritte（雷内·马格利特）的著名画作“The Treachery of Images”（图像的背叛）来说明，如下图所示。



来源：<http://collections.lacma.org/node/239578>

这幅画展示了一个烟斗和一行字“Ceci n'est pas une pipe”（这不是一个烟斗）。Magritte想要表达的观点是表现出来的烟斗并不是一个真实的烟斗。尽管事实上那个烟斗并不真实，但是任何看到这幅画的人都能很轻易地认出这张照片展示了一个烟斗，这表明观察的人的思想能够连接图画中的烟斗和观念里的烟斗，接着照片里的烟斗就和真实的那种能拿在手里的烟斗联系在一起。像这样的抽象连接是知识表达的基础，是帮助把原始的感官信息转变成有意义洞察的逻辑结构的基础。

在知识表达的过程中，计算机把原始输入的数据概括在一个模型里，这个模型是数据间结构化模式的一个显式描述。有很多种不同类型的模型，你可能对其中的一些模型比较熟悉。例如：

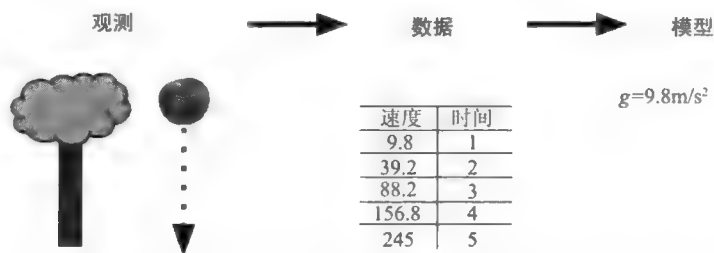
- ☐ 方程
- ☐ 像树形图和曲线图这样的图表
- ☐ 逻辑上的如果/否则（if/else）规则
- ☐ 把数据分组为类

模型的选择通常不是由机器来完成的，而是由学习的任务和所分析数据的类型来决定

的。在本章的后续部分，将会详细地讨论选择模型类型的方法。

用一个特定的模型来拟合数据集的过程称为**训练**。为什么不把它称为学习呢？第一，要牢记学习的过程不会因为数据抽象化过程的结束而终止。学习还要求更进一步的步骤，即把知识一般化从而被未来数据所用。第二，“训练”这个词更精确地描绘了模型被应用于数据的实际过程。学习是一种自下而上的归纳性推理。训练更好地表现了机器学习模型是由人类教师强加于机器学生身上的这个事实，为计算机以后想要建立的模型提供了一个体系结构。

当模型被训练后，数据转换为一个汇总了原始信息的抽象形式。必须指出的是模型并不会自己提供额外的数据，然而有时候它本身就很有趣。怎么会是这样的呢？其原因是：通过把假定的结构应用在原始数据上，它给出了不可见现象的洞察并且提供了数据是如何相互关联的理论。例如万有引力的发现过程。通过用方程来拟合观测的数据，Sir Isaac Newton（艾萨克·牛顿爵士）推导出了万有引力的概念。但是万有引力原本就是一直存在的。直到通过模型用抽象的参数定义它，特别是把它变成模型中的参数 g ，用来解释观测到的掉下来的物体，它才被作为一个概念来认识。



几个世纪以来，大多数的模型并没有导致科学观念在理论方面的发展。但是模型仍然可发现数据间原来无法看清的关系。由染色体数据所训练的模型可能会发现多个基因之间的结合导致糖尿病出现的情况；银行可能会发现一个表面上没有危害的交易类型总是有规律地出现在诈骗活动之前；心理学家可能发掘出几个特征间的结合会暗示了一个新的心理失调的出现。内在的关系是永远存在的，但是通过用不同的格式来把信息概念化，一个模型就会从一个新的视角呈现出内在的关系。

1.3.2 一般化

前面提到，直到学习者能够把抽象知识应用到将来的行动中去，学习过程才算完成。然而，在学习者继续进行之前还存在一个问题：在抽象化的过程中可能发现无数的内在关系，同时还有多种方法对这些内在关系建模。除非潜在理论的数量有所限制，否则学习者将不能利用这些信息。学习过程在一开始就会被卡住，空有大量的信息但是没有可以转化为行动的洞察。

一般化这个术语描述了把抽象化的知识转换成可以用于行动的形式。某种程度上而言，一般化是一个难以描述的模糊的过程。从传统意义上来说，它被想象成训练过程中对所有可用于数据抽象化的模型（即理论）的搜索过程。具体来说，如果你设想一个假定包含所有从数据建立的可能模型（或理论）的集合，一般化就是把这个集合里的理论的数量减少到一个

可以管理的数目。

一般而言，通过逐一检验来减少潜在概念的数量并确定哪一种理论是最有用的方法是不可行的。机器学习算法一般用一些能更快划分概念集合的捷径方法。为了这个目的，算法应用启发式方法，或者有训练地猜测在何处能找到最重要的概念。



由于启发式方法利用相近原理和其他一些经验法则，所以它不能保证找到对数据建模的最优概念集。然而，如果不利用这些捷径方法，从大的数据集中找到有用信息的任务就变得不可能了

启发式方法通常被人类用来快速地把经验推广到新的场景（new scenario）。如果你曾经在充分评估你的情况之前应用直觉来做出快速决定，那么你已经不自觉地使用了精神启发式方法。

例如，人们通过对回忆过去案例的难易程度来判断一个事件可能性的倾向，这就是启发式方法。启发式方法可能帮助解释相对汽车旅行而言，人们为什么更加普遍害怕飞行旅行这个问题，尽管统计上的结果是汽车更加危险。有关飞行旅行的事故会被高度宣传，这种令人震惊的事件更加容易被人们回想起来，然而汽车事故却鲜有刊登在报纸上。

上述例子表明，启发式方法可能会导致不合理的结论。遍览一系列常见的逻辑谬误，你可能注意到其中很多看起来是由于启发式思维导致的。例如，赌徒的谬误，也就是说相信碰到坏运气就意味着一段好运的到来。可能应用了表面上的启发式思维，错误地导致赌徒认为由于大部分随机结果是平衡的，所以所有随机结果是平衡的。

误用启发式方法的愚蠢并不局限于人类。机器学习算法应用启发式方法有时也会导致错误的结论。如果算法的结论是系统性的不精确，就说该算法有**偏差**。例如，假设一个机器学习算法学习通过寻找位于（代表嘴巴的）一条线上方两侧的两个圆，或者说眼睛，来识别人脸。这个算法就可能在识别那些不符合这个模型的人脸时出现问题，或者说产生偏差。不能识别的人脸可能包括戴眼镜、转动一定角度、侧视或者有深色皮肤的人脸。类似地，对那些浅色眼睛或者其他不同于机器所理解的世界（即其他特征的人脸），机器就会产生偏差。



现代用法中，“偏差”一词带有负面的含义。各种形式的媒体都经常宣称没有偏见，声称客观报道事件，没有受情绪左右。但是，细想一下，其实有一点偏差也许是有用的。如果没有任何的专断，在几个各有其长处和短处的相互竞争的选项中做出选择就变得困难了。确实，当今心理学领域的一些研究指出，出生时就损坏了掌管情绪的那部分大脑的人在做决定时是无所适从的，他们可能花费几个小时来纠结一些简单的决定，比如穿什么颜色的衬衫或

者在哪里吃饭。矛盾的是，有时候偏差会使我们无视一些信息，它同时又让我们能够应用一些其他的信息来行动。

1.3.3 评估学习的成功性

偏差是与任何机器学习任务的抽象化和一般化这两个过程相联系的不可避免的谬误。每一个学习者都有其弱点，并且以特定方式具有偏差；没有一个模型可以胜过所有其他模型。因此，一般化过程的最后一步就是在存在偏差的情况下判断模型的成功性。

在初始的数据集上训练模型之后，它要被一个新的数据集检验，并且判断从训练数据得到的特征推广到新数据的好坏程度。值得注意的是，一个模型完美地推广到所有未预见到的情况是极端罕见的。

在某种程度上，由于数据中的噪声或者无法解释的波动导致模型不能完美地一般化。噪声数据是由看起来随机的事件所导致的，比如：

- ❑ 测量错误，由于传感器的不精准，有时候会导致读出的数据增加一些或者减小一些。
- ❑ 报告的数据有问题，比如被调查者为了尽快完成调查问卷，就随意回答问题。
- ❑ 由于数据记录不正确而导致的错误，包括数据的缺失、空值、截断、错误编码或者取值冲突。

试图用模型拟合噪声就是称之为过度拟合问题的基础。因为根据定义，噪声是无法解释的，尝试解释噪声会导致错误的结论，该结论不能推广到新的场景中。尝试建立解释噪声的理论也会导致模型更加复杂，从而极有可能忽略了学习者努力去寻找的真实的模型。一个模型在训练时表现得很好，但是当测试时就表现很差的现象，称为过度拟合了训练数据集，即它不能很好地一般化。

过度拟合问题的解决办法视具体的机器学习方法而不同。就现在而言，重要的是要意识到问题的存在。模型处理噪声数据的好坏是判断模型成功与否的一个重要方面。

1.4 将机器学习应用于数据中的步骤

任何机器学习任务都能分解成一系列更容易管理的步骤。本书组织的步骤如下：

1) **收集数据**：无论数据是写在纸上，记录在文本文件、电子表格中或者存储在 SQL 数据库中，你都要把它转为适合分析的电子格式。数据将作为机器学习算法的学习材料，从而产生可行动的知识。

2) **探索数据和准备数据**：任何机器学习项目的质量很大程度上取决于它使用的数据的质量。机器学习过程的这个步骤一般需要大量的人工干预。一项常被引用的统计数据指出，机器学习中 80% 的努力花费在数据上。这其中的大多数时间都花费在一项称为数据探索的实践中，它要学习更多的数据信息和它们的细微差别。

3) **基于数据训练模型**：在已经准备好用于分析的数据时，你很有可能已经有了希望从数据中学习到的设想。具体的机器学习任务将会告知你选择合适的算法，算法将会以模

型的形式来表现数据。

4) **评价模型的性能**：由于每个机器模型将会产生一个学习问题的有偏差的解决方法，所以评价算法从经验中学习的优劣是很重要的。根据使用模型的类型，你应该能用一个测试数据集来评价模型的精确性，或者你可能需要针对目标应用设计模型性能的检验标准。

5) **改进模型的性能**：如果需要更好的性能，就需要利用更加高级的方法来提高模型的性能。有时候，需要完全更换为不同的模型。你可能需要补充另外的数据，或者如这个过程的第二个步骤中所做的那样，进行一些额外的数据准备工作。

在完成这些步骤以后，如果模型表现令人满意，就能将它应用到预期的任务中。根据具体的情况，为了预测（也可能是实时预测）的目的，你可能需要模型给出预测分数。例如，预测财务数据、对市场或者研究给出有用的见解，或者使诸如邮件投递或者飞机飞行之类的任务实现自动化。部署的模型无论成功或者失败，它们都可能为训练下一代的模型提供进一步的数据。

1.5 选择机器学习算法

机器学习算法的选择涉及衡量要学习数据的特征和可以使用的方法所具有的偏差。因为机器学习算法的选择很大程度上依赖于你所分析数据的类型和任务，当你在收集、探索和处理数据时就开始考虑这个过程是很有帮助的。



只要学习几个机器学习的技巧，就能把它们应用到所有的数据中的想法是很诱人的，但是要抵住这种诱惑。没有一种机器学习方法对所有环境都是最好的。这个事实被描述为**没有免费午餐定理**，由 David Wolpert 在 1996 年提出。想要了解更多的信息，请访问：<http://www.no-free-lunch.org>。

1.5.1 考虑输入的数据

所有的机器学习算法都要求输入训练数据。虽然精确的格式可能不同，但是就最基础的格式而言，输入数据是以案例 (example) 和特征 (feature) 组成的表格形式呈现的。

从名称上看，**案例**是一个被学习概念的示例性实例，它是要分析事物的最基本单位的一组数据。如果你要建立一个识别垃圾邮件的学习算法，案例就是从诸多个人电子邮件中获得的数据。想要诊断癌症肿瘤，案例可能由一些病人的活检切片组成。

短语观察单位是用来描述被测量的案例的单位。一般来说，观察单位是以交易、人、时间点、地理区域或者测量单位的形式呈现的。其他可能的观察单位包括前面所述的单位的组合，如人年，它表示同一个人在多个时间点上被记录的情况。

特征是指案例的一个属性或者特性，它们可能对学习目标概念有帮助。在前面的例子中，垃圾邮件检索数据集的特征就可能由邮箱信件所使用的文字组成。在癌症数据集中，属性可能是活检细胞染色体数据或者测得的病人的一些体征，比如体重、身高或者血压等。

下面的这个电子表格是一个**矩阵格式**的数据集，它意味着每一个案例都有相同数量的特征。在矩阵数据中，电子表格的每一行表示一个案例，每一列表示一个特征。在这个表格里，行表示汽车的案例，每列记录了汽车的不同特征，例如汽车价格、行驶的英里数、车身的颜色、变速箱（手动或者自动）。尽管在后面的章节里，你将会看到在特定的情况下偶尔会使用其他形式的数据，但矩阵格式的数据是目前机器学习中最常用的数据形式。

	A	B	C	D	E	F
1	year	model	price	mileage	color	transmission
2	2011	SEL	21992	7413	Yellow	AUTO
3	2011	SEL	20995	10926	Gray	AUTO
4	2011	SEL	19995	7351	Silver	AUTO
5	2011	SEL	17809	11613	Gray	AUTO
6	2012	SE	17500	8367	White	AUTO
7	2010	SEL	17495	25125	Silver	AUTO
8	2011	SEL	17000	27393	Blue	AUTO
9	2010	SEL	16995	21026	Silver	AUTO
10	2011	SES	16995	32655	Silver	AUTO

特征也分为很多种形式。如果一个特征所代表的特性是用数值来衡量的，那它毫无疑问称为**数值型**。另外，如果它所代表的属性是通过一组类别来表示的，这样的特征称为**分类变量**或者**名义变量**。分类变量中有一种特殊的类型：**有序变量**，它指分类变量的类别落在一个有序列表中。有序变量的例子包括衣服的尺码：小号、中号和大号，或者顾客的满意程度：用级别 1 ~ 5 来评估。考虑特征的表现形式很重要，因为数据集中特征的种类和数量有助于你找到一个适于你的任务的机器学习算法。

1.5.2 考虑机器学习算法的类型

机器学习算法可以分为两类：用来建立预测模型的有监督学习算法和用来建立描述模型的无监督学习算法。使用哪种类型的算法取决于你需要完成的学习任务。

预测模型，顾名思义，它用于这样的学习任务：利用数据集中的其他数值来预测另一个值。学习算法的目的是发现并且建模**目标特征**（需要预测的特征）和其他特征之间的关系。尽管在通常情况下，使用“预测”这个词是用来暗示对未来的预测，但是预测模型不一定是预测未来的事件，它也能用来预测过去的事情，比如通过母亲的荷尔蒙数量来预测怀孕的日期，它也可以用来实时控制高峰时段的交通信号灯。

因为预测模型对于“学什么”和“怎么学”有清晰的指导，所以训练一个预测模型的过程也称为**有监督学习**。监督并不是指需要人为干预，它是指让目标值担任监督的角色，让它告诉学习者要学习的任务是什么。具体来说，给一组数据，学习算法尝试最优化一个函数（模型）来找出属性值之间的组合方式，最终据此给出目标值。

常见的有监督学习任务是预测案例属于哪个类别，这类机器学习任务称为**分类**。不难想

象分类的潜在用途。例如，你可以预测：

- ❑ 一个足球队是会赢还是会输。
- ❑ 一个人会不会活过 100 岁。
- ❑ 一个申请人会不会贷款违约。
- ❑ 明年会不会发生地震。

被预测的目标属性是一个称为类的分类属性，它可以被分为不同的类别，这些类别称为水平。一个类能有两个或者更多的水平，水平不一定是有序的。因为分类在机器学习中广泛运用，所以有很多种类型的分类算法。

有监督学习算法也可以用来预测数值型的数据，比如收入、试验数据、考试成绩或者商品数量。为了预测这类数值型数值，能够拟合输入数据的线性回归模型是一类常见的数值型预测。尽管回归模型不是唯一的数值型模型，但至今为止它是应用最为广泛的模型。回归模型被广泛地应用于预测，因为它用表达式精确量化了输入数据和目标值之间的关系，其中包括该关系的强弱大小和不确定性。



因为可以容易地把数字转换为类别（例如，年龄在 13 ~ 19 岁之间的孩子归类为青少年）和把分类数据转化为数字（例如，把男性标示为 1，女性标示为 0），所以分类模型和数值预测模型之间的界限不太严格。

描述性模型，通过新而有趣的方式总结数据并获得洞察，从而学习任务从这些洞察中受益。与需要预测目标值的预测模型不同，在描述性模型里，没有某一个属性比其他属性更重要。事实上，因为没有要学习的目标，所以训练描述性模型的过程称为**无监督学习**。尽管思考描述性模型的应用更加困难，但是好的地方在于学习者没有特定的学习任务，这种方法在数据挖掘中经常使用。

例如，称为**模式发现**的描述性模型任务，用来识别数据之间联系的紧密性。很多时候模式发现用来对交易之间的购买数据进行**购物篮分析** (market basket analysis)。这里，目的是识别那些经常被一起购买的商品，使得学习信息能被用来改进商场的销售策略。打个比方，如果一家零售商学习到游泳裤通常会和防晒霜一起购买，那么零售商就可能把这些商品摆放得更近一些，或者做一次促销把相联系的商品捆绑销售给顾客。



模式发现最初仅仅用在零售领域，如今它开始应用在一些新的领域。例如，它能用来发现犯罪行为的模式，甄别基因缺陷或者阻止犯罪活动。

描述性模型中把数据集按照相同类型分组的任务称为**聚类**。它有时候用于细分分析，即根据相似的购买记录、捐赠记录或者人口普查信息进行人群分类，使得广告活动能够针对目标受众。尽管机器能识别各个分组，但还是需要人工介入来解释各个分组。打个比方，假设一家杂货店有 5 种不同类型的顾客分类，营销团队需要了解各种分类之间的区别以便设置最适合各种分类的促销活动。当然，这远比为每一个顾客量身定制一个推销方案容易得多。

1.5.3 为数据匹配合适的算法

下表给出了本书所讨论的机器学习算法的类型，其中每一类算法都可能有多种实现方式。尽管这些算法仅仅覆盖了所有机器学习算法的一部分，但是学习这些算法将为你提供—个充足的基础，这样你在碰到其他算法时就能够理解它们。

模型	任务	章节
有监督学习算法		
最近邻	分类	第 3 章
朴素贝叶斯	分类	第 4 章
决策树	分类	第 5 章
分类器	分类	第 5 章
线性回归	数值预测	第 6 章
回归树	数值预测	第 6 章
模型树	数值预测	第 6 章
神经网络	双重用处	第 7 章
支持向量机	双重用处	第 7 章
无监督学习算法		
关联规则	模式识别	第 8 章
k 均值聚类	聚类	第 9 章

为了给学习任务找到相应的机器学习方法，你需要从下面 4 种类型的任务之一开始，它们分别是：分类、数值预测、模式识别或者聚类。确定的学习任务将使算法的选择变得简单。例如，如果你要进行模式识别任务，可能会应用关联分析。类似地，聚类问题可能会用 k 均值算法，而数值预测则会应用回归分析或者回归树。

对于分类来说，需要把精力花费在为学习问题找到合适的分类器。这时候，考虑到不同算法间的各种差别是很有帮助的。例如，在分类问题中，决策树得到的模型就容易理解，而神经网络得到的模型则很难解释。如果你要设计一个信用评分模型，上述区别就是一个很重要的差别，因为法律要求必须告知申请者贷款申请被拒绝的原因。即使神经网络算法能更好地预测贷款违约，但是如果不能解释清楚这些预测，那么再好的预测也是没有用的。

接下来的每一章都会给出各种方法的核心优势和劣势。尽管有时候你发现这些特征在很多情况下使得某些模型被排除在考虑之外，但是模型的选择是带有随机性的。从这个角度来看，你可以自由使用那些你有信心的算法。另外，当预测准确性是主要考虑因素时，你可能需要测试多个模型，然后选择一个最好的。在后面的章节中，我们将学习利用每个模型优点的组合模型方法。

1.6 使用 R 进行机器学习

机器学习所需要的很多算法都没有包含在 R 的基本安装里。幸亏 R 是一款开源免费的软

件，机器学习算法的功能不另外收费。机器学习算法被很多对 R 软件做出贡献的专家加载到了 R 的基础配置里。那些能在用户间共享的 R 函数的集合称为**添加包**（package）。本书中所讨论的每个机器学习算法的 R 免费添加包都是已经存在的。事实上，本书中仅仅讨论了普遍运用的机器学习添加包中很小的一部分。

如果你对 R 添加包的广度感兴趣（在写作本书时，共有 4209 个可用的添加包），你可以游览 Web 或者 FTP 服务器上的 **R 综合文档网络**（Comprehensive R Archive Network, CRAN），这些服务器位于全球各地，它们提供最新版本的 R 软件和 R 添加包以供下载。

如果你的 R 软件是下载得到的，很可能你就是从 CRAN 下载的。CRAN 网站可以由此进入：

<http://cran.r-project.org/index.html>。



如果你没有安装 R，CRAN 网站提供了 R 安装说明，给出了遇到困难时在何处可以获得帮助的信息。

通过页面左边的 **Packages** 链接，你可以浏览按照字母顺序排列或者根据发布日期排序的添加包列表。也许更好的是，**CRAN Task Views**（任务视图）提供了一个根据学科分类整理的添加包列表。机器学习的任务视图列出了本书所包含的添加包（和其他更多的添加包），可以由此进入：

<http://cran.r-project.org/web/views/MachineLearning.html>。

安装和加载 R 添加包

尽管有大量可用的 R 添加包，但添加包的格式实际上使得它的安装和使用变成一个很省力的过程。为了举例说明添加包的应用，我们将安装和加载由 Kurt Hornik、Christian Buchta 和 Achim Zeileis 开发的 RWeka 添加包（想得到更多的信息，参考杂志《Computational Statistics》第 24 期第 225 ~ 232 页的文章“Open-Source Machine Learning: R Meets Weka”）。RWeka 添加包提供了一个基于 Java 平台的 R 能够使用的机器学习算法的函数集合，该集合由 Ian H. Witten 和 Eibe Frank 开发。关于 Weka 的详细信息，参考：

<http://www.cs.waikato.ac.nz/~ml/weka/>。



为了使用 RWeka 添加包，如果没有 Java 软件，你将需要预先安装（很多计算机买来的时候预先安装了 Java）。Java 是一组编程工具，可以免费使用，它允许使用跨平台的应用，比如 Weka。想要了解更多信息，或者为你的系统下载 Java，可以访问：<http://java.com>。

1. 安装 R 添加包

安装添加包的最直接方式就是通过 `install.packages()` 函数。要安装 RWeka 添加

包，在 R 命令提示符后输入：

```
> install.packages("RWeka")
```

R 接下来就会连接到 CRAN 并且下载与你的操作系统相匹配的添加包。一些像 RWeka 这样的添加包在它们能够使用之前要求先安装额外的添加包（这些额外的添加包称为依赖添加包（dependency））。默认情况下，安装程序会自动下载并安装所有的依赖添加包。



当你第一次安装添加包时，R 可能会让你选择一个 CRAN 镜像。如果出现了这个对话框，那么就选择地理位置离你最近的镜像。这样一般会提供最快的下载速度。

默认的安装选项对大多数系统是合适的。然而，有时你可能想把添加包安装到其他的位置。例如，如果你没有系统的超级用户权限或者管理员权限，你可能需要指定其他安装路径。这可以通过 lib 选项来完成，如下所示：

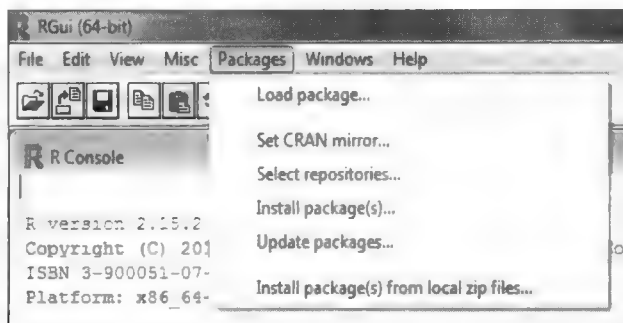
```
> install.packages("RWeka", lib="/path/to/library")
```

安装函数也提供其他多个安装选项：从本地文件安装、从源文件安装或者使用测试版本。你可以用下面的命令在帮助文件中阅读这些选项。

```
> ?install.packages
```

2. 用图形界面来安装

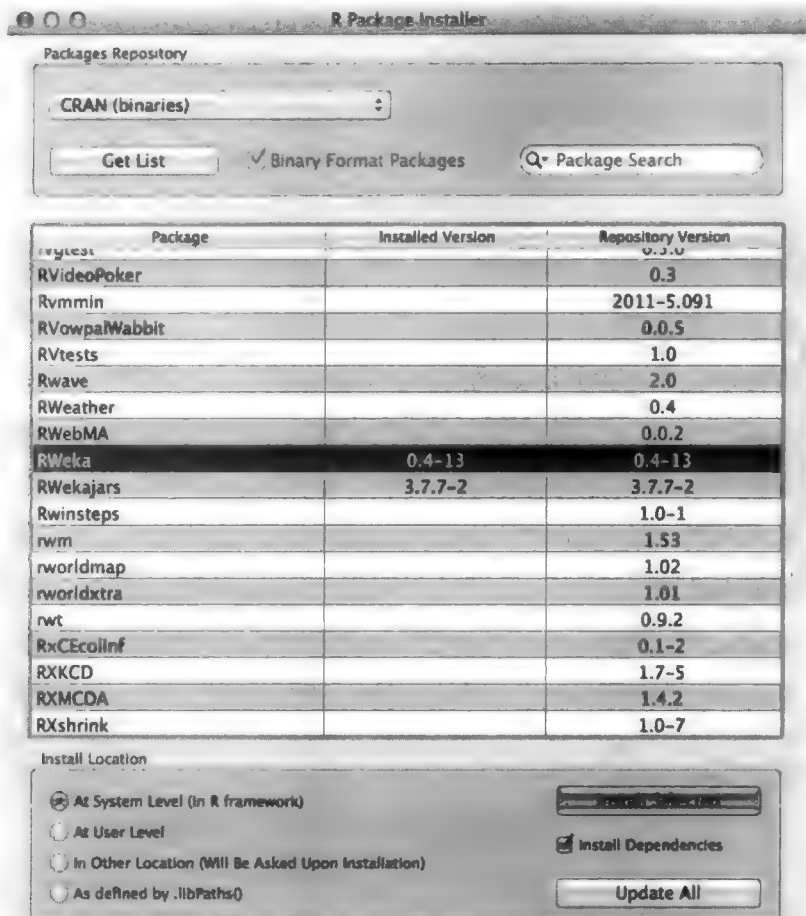
作为输入 `install.packages()` 命令的另外一种替代方法，R 提供了图形用户界面（GUI）来安装添加包。在微软的 Windows 操作系统中，可以用 **Packages** 菜单下的 **Install package(s)** 子命令来完成，如下图所示。在 Mac OS X 系统中，这个命令是在 **Packages & Data** 菜单下标记为 **Package Installer** 的子命令。



在 Windows 系统中，在启动 **Install package** 后（如果没有选择过 CRAN 镜像，就弹出一个窗口让你选择 CRAN 镜像），一个添加包的长列表将会出现。只要把鼠标滚动到 RWeka 添加包的位置，然后单击下方的 **OK** 按钮，就会把该添加包以及所有的依赖添加包都安装在默认的位置。

在 Mac OS X 系统中，**Package Installer** 菜单提供了额外的选项。为了载入添加包列表，

可以单击 Get List 按钮，将鼠标滚动到 RWeka 添加包（或者使用 Package Search 功能），再单击 Install Selected。注意，在默认的情况下，除非选中 Install Dependencies 复选框，否则 Mac OS X 的添加包安装程序是不会安装依赖添加包的，如下图所示。



3. 载入 R 添加包

为了节约内存，R 默认不会载入每个已安装的添加包。当用户需要某个添加包时，只要用 `library()` 函数把该包载入 R 即可。



`library` 这个函数名使得它容易与 `package` 相互搞混。可是，准确来说，`library` 指的是安装添加包的位置，而不是添加包本身。

为了加载先前我们安装的 RWeka 添加包，可以输入下面命令：

```
> library(RWeka)
```

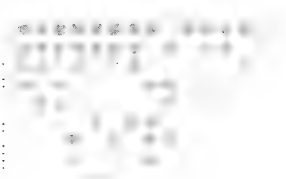
除了 RWeka 包以外，在后面的章节中我们还将用到其他几个添加包。当使用它们时，我们会提供相应的安装说明。

1.7 总结

机器学习起源于统计学、数据库科学和计算机科学的交互。它是一个强大的工具，能够在大量的数据中找到可行动的洞察。然而，人们仍需持谨慎的态度，避免现实生活中机器学习的普遍滥用。

从概念上讲，机器学习涉及把数据抽象为结构化表示，并把这个结构化表示进行一般化，然后推广到行动中。从更加实用的角度讲，机器学习者使用包含所学习概念的案例和特征的数据，然后把这个数据概括成一个模型的形式，接着该模型就被用作预测或者描述的目的。这些还能被细分为具体的任务，包括分类、数值预测、模式识别和聚类。在大量的选择中，机器学习算法都是以输入数据和学习任务为基础进行算法选择的。

R 通过 R 社区作者编写的添加包来为机器学习提供支持。这些强大工具的下载是免费的，但是要在使用之前先安装它们。在第 2 章中，将继续介绍用于为机器学习管理和准备数据的基本 R 命令。



Chapter 2 第 2 章

数据的管理和理解

任何机器学习项目初期的核心部分都是与管理 and 理解所收集的数据有关的。尽管这些工作不像建立和部署模型那样令人有成就感（建立和部署模型阶段就开始看到了劳动的收获），但是你不能忽视这些准备工作。

任何学习算法的好坏取决于输入数据的好坏。在很多情况下，输入数据是复杂的、凌乱的并且取自多种不同渠道和格式。因为这些复杂性，所以投入到机器学习项目中的很大一部分精力要花在数据准备和探索过程中。

本章分成 3 节。第一节讨论 R 用来存储数据的基本数据结构。学完这一节后，你创建和管理数据集时，就对这些数据结构非常熟悉了。第二节是实践，这节讨论了从 R 中输入或者输出数据的几种常用函数。第三节通过介绍一个探索真实数据集的过程来说明理解数据的方法。

学完本章后，你将理解：

- 基本的 R 数据结构以及如何使用它们来存储和提取数据。
- 如何把不同来源格式的数据导入 R。
- 理解并可视化复杂数据的常用方法。

因为 R 考虑数据的方式定义了你应该考虑数据的方式，所以在进入数据准备的工作之前，理解基本的 R 数据结构是很有帮助的。然而，如果你已经对 R 的数据结构很熟悉了，你完全可以跳过这部分，直接学习数据预处理部分。

2.1 R 数据结构

在程序语言中有很多种形式的数据结构，在应用到特定的任务时，它们各有优势和劣势。因为 R 是一个在统计数据分析中广泛运用的程序语言，所以 R 所用的数据结构的设计目

的是使它易于处理这类任务的数据。在机器学习中经常使用的 R 数据结构是：向量、因子、列表、数组和数据框。每一种数据类型都针对一类具体的数据管理任务，所以知道它们是如何与 R 项目相互交互是至关重要的。

2.2 向量

R 的基本数据结构是**向量**。向量存储一组有序的值，称为**元素**。一个向量可以包含任意数量的元素。然而，所有的元素必须是一样的类型，比如，一个向量不能同时包含数字和文本。

在机器学习中常用的几种向量类型包括：`integer`（整型，没有小数位的数字）、`numeric`（数值型，包含小数的数字）、`character`（字符型，文本数据）或者 `logical`（逻辑型，取值为 `TRUE` 或者 `FALSE`）。还有两个特殊的值：`NULL`，用来表明没有任何值；`NA`，用来表明一个缺失值。

手工输入大量的数据会单调乏味，但是一些简单的向量还是可以用组合函数 (`combine function`) `c()` 来创建。向量也能通过使用箭头运算符 “`<-`” 来给它赋一个名字，这是 R 的赋值操作符，与很多其他程序语言中的赋值操作符 “`=`” 的使用方法差不多。

例如，我们构建一个包含 3 个体检病人数据的向量。创建一个字符型向量 `subject_name`，它包含 3 个病人的姓名；一个数值型向量 `temperature`，它包含每个病人的体温；以及一个逻辑型向量 `flu_status`，它包含每个病人的诊断情况，如果病人患有流感取值为 `TRUE`，否则为 `FALSE`。3 个向量如下所示：

```
> subject_name <- c("John Doe", "Jane Doe", "Steve Graves")
> temperature <- c(98.1, 98.6, 101.4)
> flu_status <- c(FALSE, FALSE, TRUE)
```

因为 R 中的向量有固有的顺序，所以其数据能通过计算向量中各元素的序号来访问，序号是从 1 开始算起的，并且在向量名字的后面用方括号括起这个序号（例如，`[` 和 `]`）。例如，为了获得温度向量中 Jane Doe 或者序号为 2 的病人的体温，只要简单地输入：

```
> temperature[2]
[1] 98.6
```

为了从向量中提取数据，R 提供了各种方便的方法。一个范围内的值可以通过冒号操作符获得。例如，为了获得 Jane Doe 和 Steve Graves 的体温，输入：

```
> temperature[2:3]
[1] 98.6 101.4
```

通过指定一个负的序号可以把该项排除在输出数据之外。要想排除 Jane Doe 的体温数据，输入：

```
> temperature[-2]
[1] 98.1 101.4
```

最后，可以通过一个逻辑型向量来标识每一项是否包含在内，有时候这也是很有用的。

例如，需要包括前两个温度读数，但是排除第三个，就可以输入：

```
> temperature[c(TRUE, TRUE, FALSE)]
[1] 98.1 98.6
```

不久你将会看到，向量是很多其他 R 数据结构的基础。因此，懂得不同类型的向量操作对在 R 中操作数据是很重要的。



下载实例代码你能用你的账号在网站 <http://www.packtpub.com> 上下载你已经购买的所有 Packt 出版社的书的实例代码文件。如果你是在其他地方购买这本书的，你可以通过访问 <http://www.packtpub.com/support> 网站并且注册，可以通过电子邮件把文件直接发给你。

2.3 因子

如果你回忆第 1 章的内容，用类别值来代表特征的属性称为**名义属性**。尽管可以用一个字符型向量来存储名义属性数据，但 R 提供了称为**因子**（factor）的专用数据结构来表示这种属性数据。其实因子是向量的一个特例，它单独用来标识名义属性（或变量）。在上面构建的医学体检数据集中，我们可以用一个因子来表示 gender（性别），因为它有两个类别：MALE 和 FEMALE。

为什么不用 character（字符型）向量呢？使用因子的一个优势在于：因为类别标签只存储一次，所以它们一般比字符型向量更有效。不像字符型向量需要存储 MALE、MALE、FEMALE，在使用因子时，计算机只要存储 1、1、2，这样可以节约内存。另外，一些机器学习算法是用特别的程序来处理分类变量的，把分类变量编码为因子可以确保模型能够合理地处理这类数据。

要把字符型向量转换成因子，只需要应用 factor() 函数。例如：

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE    FEMALE  MALE
Levels: FEMALE MALE
```

注意，当性别数据显示出来以后，R 会输出额外的信息来表明性别因子的水平。水平由数据可能取到的所有类别组成，在这个例子中是 MALE 或者 FEMALE。

当创建因子以后，可以加入另外的没有在数据中出现的水平。假设我们用另一个因子表示血型变量，如下所示：

```
> blood <- factor(c("O", "AB", "A"),
                  levels = c("A", "B", "AB", "O"))
> blood
[1] O  AB  A
Levels: A B AB O
```

注意，当我们为 3 个病人定义 blood（血型）因子时，我们用 levels = 语句来给出一

个额外的向量，该向量给出了 4 个可能的血型。这样，即使数据仅包含 O 型、AB 型和 A 型，但输出“Levels: A B AB O”指出，所有的 4 种血型和血型因子 blood 存储在一起。存储额外的水平使未来增加其他血型类型成为可能。它也保证了尽管血型 B 没有被记录在我们的数据中，但是当我们创建血型类型表时，我们能知道类型 B 是存在的。

2.3.1 列表

另一种特殊类型的向量——列表，它用来存储一组有序的值。然而，不像向量要求所有的元素都必须是同一种类型，列表允许收集不同类型的值。由于这个灵活性，列表一直用于存储不同类型的输入和输出数据，以及存储机器学习模型所使用的结构参数。

例如，考虑我们构建的体检病人的数据集，3 个病人的数据存储在 5 个向量中。如果我们要显示 John Doe（对象 1）所有的数据，我们需要输入 5 条 R 命令：

```
> subject_name[1]
[1] "John Doe"
> temperature[1]
[1] 98.1
> flu_status[1]
[1] FALSE
> gender[1]
[1] MALE
Levels: FEMALE MALE
> blood[1]
[1] O
Levels: A B AB O
```

显示一个病人的体检数据看上去像是一个庞大的工程。列表结构使我们能够把所有病人的数据放到一个对象里，使我们能够重复使用。

与使用 `c()` 创建一个向量相类似，列表的创建使用 `list()` 函数，与下面例子中显示的一样。一个明显的不同是，当列表建立以后，你可以选择给列表中每一项的值命名（与下面例子中的 `fullname` 一样）。名字不是必需的，但是它使得接下来能够通过名字来访问列表中的值，而不是像向量那样通过位置序号。

```
> subject1 <- list(fullname = subject_name[1],
                   temperature = temperature[1],
                   flu_status = flu_status[1],
                   gender = gender[1],
                   blood = blood[1])
```

现在输出一个病人的数据仅需要输入下列简单的一条命令：

```
> subject1
$fullname
[1] "John Doe"

$temperature
[1] 98.1
```

```

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: FEMALE MALE

$blood
[1] O
Levels: A B AB O

```

注意，值是由前面命令中指定的名字标识的。尽管列表也能用访问向量的方法来访问，但是名字使得访问列表值变得更加明确。如下面所示，它不需要记住温度所在的位置：

```

> subject1[2]
$temperature
[1] 98.1

```

通过在列表名后面附加一个“\$”符号和值的名字，直接访问温度会变得很简单：

```

> subject1$temperature
[1] 98.1

```

如果在列表中增加或者移除某些值，通过名字来访问值的方式能保证在列表序号改变以后，不会取回错误的列表项。

也可以通过指定一个名字向量来获取列表中的多个列表项：

```

> subject1[c("temperature", "flu_status")]
$temperature
[1] 98.1

$flu_status
[1] FALSE

```

尽管整个数据集能用一个列表（或者列表的列表）来构建，但是由于构建数据集是如此常用，所以 R 专门为这个任务提供了一种特殊的数据结构（即数据框）。

2.3.2 数据框

到目前为止，机器学习中使用的最重要的 R 数据结构就是数据框。因为它既有行数据又有列数据，所以它是一个与电子表格或数据库相类似的结构。在 R 术语中，数据框定义为一个向量列表或者因子列表，每一列都有相同数量的值。因为数据框准确来说是一个向量列表，所以它结合了向量和列表两个方面的特点。

下面为前面用到的病人数据集构建一个数据框。这里我们使用前面创建的病人数据向量，`data.frame()` 函数把它们组合成一个数据框：

```

> pt_data <- data.frame(subject_name, temperature, flu_status,
  gender, blood, stringsAsFactors = FALSE)

```

你可能在上述代码中注意到一些新的东西。它加入了一个新的参数 `stringAsFactors = FALSE`。如果不指定这个选项，R 将会自动把每个字符向量转化为因子。这个参数有时

候是有用的，但有时候又会显得累赘。例如，这里，`subject_name` 字段显然不是分类数据，其中的取值也不是分类变量的类别。因此，仅仅在任务需要的时候，我们才会设置 `stringsAsFactors` 选项为 `TRUE`，使字符转化成因子。

当我们显示 `pt_data` 数据框时，可以看到它的结构与先前使用的数据结构略有不同：

```
> pt_data
  subject_name temperature flu_status gender blood
1   John Doe      98.1 FALSE      MALE      O
2   Jane Doe      98.6 FALSE     FEMALE     AB
3 Steve Graves    101.4  TRUE      MALE      A
```

与一维向量、因子和列表相比，数据框是两维的，因此它显示为矩阵格式。在数据框中，病人的每个数据向量为—列，每个病人的数据是一行。用机器学习术语来讲，列代表的是特征或属性，行代表的是案例。

为了提取整列（即整个向量）数据，利用数据框就是向量列表这一事实。与列表相类似，提取一个单独的元素最直接的方法（这里是提取一个向量或者—列数据），是通过名字来引用它。例如，为了提取 `subject_name` 向量，输入如下命令：

```
> pt_data$subject_name
[1] "John Doe"      "Jane Doe"      "Steve Graves"
```

与列表相类似，可以用名称向量从一个数据框中提取多列数据：

```
> pt_data[c("temperature", "flu_status")]
  temperature flu_status
1      98.1 FALSE
2      98.6 FALSE
3     101.4  TRUE
```

当我们通过这种方式访问数据框时，输出的结果还是一个数据框，它包含目标列所有行的数据。你也可以输入 `pt_data[2:3]` 来提取 `temperature` 和 `flu_status` 列，但是通过名字来列出所要求的列将会使 R 代码清晰、容易修改。

为了提取数据框中的值，我们可以用前面学过的访问向量中值的方法，但是有一个很重要的不同。因为数据框是两维的，所以它需要你指定所要提取的数据的行和列。格式为 “[`rows,columns`]”，先指定行号，接着是一个逗号，再指定列号，行号和列号都是从 1 开始的。

例如，为了提取病人数据框中第一行、第二列的值（John Doe 的体温值），可以输入：

```
> pt_data[1, 2]
[1] 98.1
```

如果需要提取多于一行或者—列的数据，可以指定所需要的数据所对应的行号向量和列号向量来完成。下面的语句将从第 1、3 行以及第 2、4 列中提取数据：

```
> pt_data[c(1, 3), c(2, 4)]
  temperature gender
1      98.1     MALE
3     101.4     MALE
```

要提取所有行或者列，不用一一列举，只要让行或者列的部分空白就行了。例如，提取第一列中所有行的数据：

```
> pt_data[, 1]
[1] "John Doe"      "Jane Doe"      "Steve Graves"
```

提取第一行中所有列的数据：

```
> pt_data[1, ]
  subject_name temperature flu_status gender blood
1   John Doe      98.1      FALSE   MALE      O
```

提取所有数据：

```
> pt_data[, ]
  subject_name temperature flu_status gender blood
1   John Doe      98.1      FALSE   MALE      O
2   Jane Doe      98.6      FALSE  FEMALE      AB
3 Steve Graves    101.4       TRUE    MALE      A
```

前面学习的访问列表和向量中值的方法也可以用来提取数据框的行和列。例如，列除了能通过位置访问外，也能通过名称访问，并且负号也能用来排除特定行或者列的数据。因此，命令：

```
> pt_data[c(1, 3), c("temperature", "gender")]
```

等价于：

```
> pt_data[-2, c(-1, -3, -5)]
```

为了熟练运用数据框，可以尝试用前面的病人数据来练习这些操作。或者，如果你自己的数据集进行练习就更好了。这些操作类型对我们以后将学习的章节是很重要的。

2.3.3 矩阵和数组

除了数据框以外，R 提供了用于存储表格形式数据的专用数据结构。**矩阵**是一种表示行和列数据的两维表格数据结构。R 矩阵能包含任何一种单一类型的数据，但是大多数情况下矩阵是用来做数学运算的，因此矩阵通常存储数值型数据。

要想创建一个矩阵，仅需要向 `matrix()` 函数提供一个数据向量，紧跟着用一个参数指定行数 (`nrow`) 或者列数 (`ncol`)。例如，要想创建一个 2×2 的矩阵，用于存储字母表前 4 个字母，使用 `nrow` 参数要求数据分为两行：

```
> m <- matrix(c('a', 'b', 'c', 'd'), nrow = 2)
> m
      [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
```

这个与用 `ncol = 2` 产生的矩阵是等价的：

```
> m <- matrix(c('a', 'b', 'c', 'd'), ncol = 2)
> m
```

```
      [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
```

你将会注意到 R 先加载矩阵的第一列，然后加载第二列。这称为**按列顺序**。为了进一步说明这个概念，观察当我们在矩阵中加入更多值以后会发生什么。

一共有 6 个值，设定 `nrow` 参数为 2，这将创建一个具有 2 行 3 列的矩阵：

```
> m <- matrix(c('a', 'b', 'c', 'd', 'e', 'f'), nrow = 2)
> m
      [,1] [,2] [,3]
[1,] "a"  "c"  "e"
[2,] "b"  "d"  "f"
```

类似地，设定 `ncol` 参数为 2，这将创建一个具有 3 行 2 列的矩阵：

```
> m <- matrix(c('a', 'b', 'c', 'd', 'e', 'f'), ncol = 2)
> m
      [,1] [,2]
[1,] "a"  "d"
[2,] "b"  "e"
[3,] "c"  "f"
```

与数据框一样，矩阵中的值也能用 “[rows, column]” 这样的方式来提取。例如，`m[1, 1]` 返回值 `a`，`m[3, 2]` 从矩阵 `m` 中提取值 `f`。类似地，也可以提取矩阵的整行或者整个列，例如：

```
> m[1, ]
[1] "a" "d"
> m[, 1]
[1] "a" "b" "c"
```

与矩阵结构非常接近的是**数组 (array)**，它是一个多维数据表。一个矩阵含有行和列值；一个数组包含行、列以及任意多层的值。尽管在后面的章节中我们偶尔会使用矩阵，但是数组的使用就超出了本书的学习范围。

2.4 用 R 管理数据

当处理大量数据集时，面临的挑战包括从各种不同来源中收集、准备和管理数据。R 提供了从诸多常见格式的数据源加载数据的工具，它们使得这些任务变得容易。

2.4.1 保存和加载 R 数据结构

当你花费了很长时间把某种数据框转换成你所需要的数据结构时，你不必每次重新打开 R 会话，从头开始重复前面的工作。要想把一种特定的数据结构保存到一个文件里，使它以后能被重新加载或者把这种数据结构转移到另一个系统里，可以用 `save()` 函数。`save()` 函数把 R 数据结构写到由 `file` 参数设定的位置。R 数据文件有一个文件扩展名 `.RData`。

如果我们有 3 个对象 `x`、`y` 和 `z`，可以用下面的命令把它们保存到文件 `mydata.RData` 中：

```
> save(x, y, z, file = "mydata.RData")
```

无论 `x`、`y` 和 `z` 是向量、因子、列表或者数据框，它们都会保存到文件 `mydata.RData` 里。`load()` 命令将会加载任何一种保存在以 “.RData” 为文件扩展名的文件中的数据结构。为了加载我们在先前代码中保存的 “`mydata.RData`” 文件，只需要输入：

```
> load("mydata.RData")
```

这将会重新创建 `x`、`y` 和 `z` 数据结构。



要特别小心你正在加载的数据结构！你用 `load()` 命令导入的文件中所存储的所有数据结构都会加载到你的工作区，即使它们会覆盖工作区中其他一些你正在用的东西。

如果你需要立即结束当前的 R 会话，`save.image()` 命令将会把你所有的会话写入一个叫做 `.RData` 的文件里。默认情况下，R 将会在下次启动时寻找这个文件，上次 R 结束时的 R 会话将会重现，就像你最后离开 R 时一样。

2.4.2 用 CSV 文件导入和保存数据

公开的数据通常存储在文本文件中。文本文件几乎可以在所有的计算机和操作系统中阅读，这种格式几乎全球通行。由于 Microsoft Excel 类的电子表格数据操作容易便捷，所以文本格式文件也能从此类程序中导入或者导出。

表格数据文件 (tabular 或者 table) 采用矩阵形式的结构，这种形式数据的每一行表示一个案例，每个案例有相同数量的特征。每一行的特征值由一个预先定义的称为**分隔符**的符号来区分。通常情况下，表格数据文件的第一行给出数据每一列的名称。该行称为**标题行**。

最常用的表格文本文件格式可能是**逗号分隔值** (Comma-Separated Value, CSV) 文件。根据名字可知，这种文件格式使用逗号作为分隔符。CSV 文件能在很多常用的应用程序内导入和导出。一个表示先前构建的医疗体检数据集的 CSV 文件就像下面这样：

```
subject_name,temperature,flu_status,gender,blood_type
John Doe,98.1,FALSE,MALE,O
Jane Doe,98.6,FALSE,FEMALE,AB
Steve Graves,101.4,TRUE,MALE,A
```

要想把这个 CSV 文件加载到 R 中，使用 `read.csv()` 函数，如下所示：

```
> pt_data <- read.csv("pt_data.csv", stringsAsFactors = FALSE)
```

给定 R 工作目录下的一个病人数据文件 `pt_data.csv`，上面的命令将会把 CSV 文件读入到名为 `pt_data` 的数据框。就像先前在构建数据框时那样，我们需要使用 `stringsAsFactors = FALSE` 这个参数来阻止 R 把所有的文本变量转换成因子。这个转换的步骤最好是由你而不是由 R 来操作。

如果数据是在 R 工作目录之外，你要详细列出 CSV 文件的路径。例如，当调用 `read.`

`csv()` 函数时要输入 `/path/to/mydata.csv`。

默认情况下, R 假设 CSV 文件包含一个标题行, 标题行列出了数据集内特征的名字。如果一个 CSV 文件没有标题行, 要指定选项 `header = FALSE`, 就像下面命令显示的那样, R 用 `V1`、`V2` 等默认值来指定属性名:

```
> mydata <- read.csv("mydata.csv", stringsAsFactors = FALSE,
header = FALSE)
```

`read.csv()` 函数是 `read.table()` 函数的一个特例。`read.table()` 函数能读取具有多种不同格式的表格数据, 包括其他的分隔形式, 比如制表符分隔的值 (Tab-Separated Value, TSV)。要想了解更多关于 `read.table()` 函数族的信息, 用命令 `?read.table` 来查询 R 的帮助页面。

要想把一个数据框保存成 CSV 文件, 需要使用 `write.csv()` 函数。如果数据框名是 `pt_data`, 只需要输入:

```
> write.csv(pt_data, file = "pt_data.csv")
```

这就会把一个名为 `pt_data.csv` 的文件保存到 R 工作目录中。

2.4.3 从 SQL 数据库导入数据

如果数据存储在一个 ODBC SQL (Open Database Connectivity, ODBC ; Structured Query Language, SQL) 数据库中, 比如 Oracle、MySQL、PostgreSQL、Microsoft SQL 或者 SQLite。Brian Ripley 创建的 RODBC 添加包可以把这类数据直接导入 R 数据框中。

不论是何种操作系统还是数据库管理系统 (Database Management System, DBMS), ODBC 都是一个连接到数据库的标准规范。如果你先前有通过 ODBC 连接到数据库的经验, 你很有可能是通过数据源名称 (Data Source Name, DSN) 连接的。要想使用 RODBC 添加包, 你需要 DSN, 加上一个用户名和密码 (如果数据库需要)。



配置 ODBC 连接的指令与特定的操作系统和 DBMS 相结合的方式高度相关。如果你在设定 ODBC 连接时遇到困难, 就需要与数据库管理员确认原因。另一种获得帮助的方式是通过 RODBC 添加包的附加文档 (vignette), 你能在 R 中用命令 `print(vignette("RODBC"))` 来访问。

如果你还没有安装该添加包, 需要先安装并且加载 RODBC 添加包:

```
> install.packages("RODBC")
> library(RODBC)
```

接下来, 我们将打开一个到 DSN 为 `my_dsn` 的数据库的连接, 该连接的名称为 `mydb`:

```
> mydb <- odbcConnect("my_dsn")
```

另外, 如果你的 ODBC 连接需要用户名和密码, 在调用 `odbcConnect()` 函数时应该给出它们。

```
> mydb <- odbcConnect("my_dsn", uid = "my_username"
  pwd = "my_password")
```

既然我们有一个开放的数据库连接，就能够用 `sqlQuery()` 函数执行 SQL 查询得到数据库的行数据，从而建立 R 数据框。这个函数与很多创建数据框的函数一样，允许我们指定 `stringsAsFactors = FALSE` 来阻止 R 把字符型数据转换成因子。

`sqlQuery()` 函数使用典型的 SQL 查询语言，如下所示：

```
> patient_query <- "select * from patient_data where alive = 1"
> patient_data <- sqlQuery(channel = mydb, query = patient_query,
  stringsAsFactors = FALSE)
```

得到的 `patient_data` 变量是一个数据框，它包含使用 SQL 查询选择的所有存储在 `patient_query` 中的数据行。

如果你已经结束使用数据库，用下面的命令关闭 ODBC 连接：

```
> odbcClose(mydb)
```

这将会关闭 `mydb` 连接。尽管 R 会在 R 会话结束后自动关闭 ODBC，但这样明显关掉连接是一个好习惯。

2.5 探索和理解数据

在收集数据并且把它们加载到 R 数据结构以后，机器学习的下一个步骤是仔细检查数据。在这个步骤里，你将开始探索数据的特征和案例，并且找到数据的独特之处。你对数据的理解越深刻，你将会更好地让机器学习模型匹配你的学习问题。

理解数据探索的最好方法就是通过实例。在本节中，我们将探索 `usedcars.csv` 数据集，其中包含在流行的美国网站上最近发布的关于二手车打折销售广告的真实数据。



`usedcars.csv` 数据集能在 Packt 网站下载。如果你要和例子一起操作，要确保这个文件下载并且保存在你的 R 工作目录里。

因为数据集存储为 CSV 形式，所以我们能用 `read.csv()` 函数把数据加载到 R 数据框中：

```
usedcars <- read.csv("usedcars.csv", stringsAsFactors = FALSE)
```

有了 `usedcars` 数据框，现在我们将担任数据科学家的角色，任务是理解二手车数据。尽管数据探索是一个不确定的过程，但可以把这个步骤想象成一个调查过程，在这个步骤里回答关于数据的问题。具体的问题可能会因任务的不同有所变化，但是问题的类型一般是相似的。不管数据集的大小，你应该能把这个调查的基本步骤应用到任何你感兴趣的数据集中。

2.5.1 探索数据的结构

调查中第一个问题应该是数据是怎么组织的。如果你足够幸运，数据源会提供一个**数据字典**，它是一个描述数据特征的文档。在我们的例子里，二手车数据并不包含这个文件，所以我们需要创建我们自己的数据字典。

函数 `str()` 提供了一个显示数据框结构的方法，或者说它提供了显示所有同时包含向量和列表的数据结构的方法。这个函数可以用来创建数据字典的基本轮廓：

```
> str(usedcars)
'data.frame':150 obs. of 6 variables:
 $ year      : int  2011 2011 2011 2011 ...
 $ model     : chr  "SEL" "SEL" "SEL" "SEL" ...
 $ price     : int  21992 20995 19995 17809 ...
 $ mileage   : int  7413 10926 7351 11613 ...
 $ color     : chr  "Yellow" "Gray" "Silver" "Gray" ...
 $ transmission: chr  "AUTO" "AUTO" "AUTO" "AUTO" ...
```

对于这样一条简单的命令，我们就知道了关于数据集的大量信息。语句 `150 obs` 告诉我们数据一共包含 150 个观测值或者案例。观测值的数量一般简写为 n 。因为我们知道数据描述的是二手车，所以现在可以认为供销售的车有 $n=150$ 辆。

语句 `6 variables` 指的是数据中记录了 6 个特征。这些特征根据名字排列成独立的行。查看特征 `color` 所在的那一行，我们注意到一些额外的信息：

```
$ color      : chr  "Yellow" "Gray" "Silver" "Gray" ...
```

在变量名的后面，`chr` 告诉我们这个特征是字符型的。在这个数据集中，3 个变量是字符型，另外 3 个注明是 `int`，表明是整数型。尽管这个数据集仅包含整数型和字符型，但当使用非整数型数据时，你还可能会碰到数值型 `num`（例如，含有小数点的数字）。所有因子都会列为 `Factor` 型。在每个变量类型的后面，R 给出这个特征的最前面的几个值。值 `"Yellow"`、`"Gray"`、`"Silver"` 和 `"Gray"` 是 `color` 特征的前 4 个值。

根据相关领域知识，特征名称和特征值可以使我们对变量所代表的含义做出假定。变量 `year` 可能指汽车制造的时间，也可能指汽车广告贴出的时间。接下来我们将更加仔细地调查这个特征，因为它的 4 个案例值（2011, 2011, 2011, 2011）适用于上述任何一个可能性。变量 `model`、`price`、`mileage`、`color` 和 `transmission` 极有可能指的是销售汽车的特征。

尽管数据似乎被赋予了有内在含义的变量名，但实际应用中并不是所有的数据都是这样的。有时候，数据集的特征可能是没有具体含义的名字、代号或者像 `V1` 这样的简单数字。通过进一步调查，从而确定特征名称确切代表的含义是必不可少的。然而，即使特征名称有具体的意义，也要谨慎检查提供给你的标签含义的正确性。我们继续进行下面的分析。

2.5.2 探索数值型变量

为了调查二手车数据中的数值型变量，我们将使用一组普遍使用的描述数值的指标，它们称为**汇总统计量**。`summary()` 函数给出了几个常用的汇总统计量。我们看看二手车数据

中的 year 特征:

```
> summary(usedcars$year)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2000   2008   2009   2009   2010   2012
```

即使你对汇总统计量不熟悉,你也能从 `summary()` 函数输出结果的标题中猜出一些。现在先不管特征 year 所代表的具体含义,事实上,当我们看到诸如 2000、2008 以及 2009 这样的数字时,我们相信变量 year 表示汽车制造的时间而不是汽车广告打出的时间,因为我们知道汽车是最近才挂牌出售的。

我们也能使用 `summary()` 函数同时得到多个数值型变量的汇总统计量:

```
> summary(usedcars[c("price", "mileage")])
      price      mileage
Min.   : 3800  Min.   : 4867
1st Qu.:10995  1st Qu.: 27200
Median :13592  Median : 36385
Mean   :12962  Mean   : 44261
3rd Qu.:14904  3rd Qu.: 55125
Max.   :21992  Max.   :151479
```

`summary()` 函数提供的 6 个汇总统计量是探索数据的简单但强大的工具。汇总统计量可以分为两种类型:数据的中心测度和分散程度测度。

1. 测量中心趋势——平均数和中位数

中心趋势测度是这样一类统计量,它们用来标识一组数据的中间值。你应该已经熟悉常用的一个测量中心趋势的指标——平均数。在一般使用中,当一个数被认为是平均数时,它是落在总体样本的两个极值之间的某个位置。一个中等学生的成绩可能落在他的同学之间;一个平均体重不会是特别重或者特别轻。平均数是具有代表性的,它和组里的其他值不会差得太多。你可以把它设想成一个所有其他值用来进行参照的值。

在统计学中,平均数也叫做均值,它定义为所有值的总和除以值的个数。例如,要想计算收入分别是 \$36 000、\$44 000 和 \$56 000 的 3 个人的平均收入,我们可以如下计算:

```
> (36000 + 44000 + 56000) / 3
[1] 45333.33
```

R 也提供一个 `mean()` 函数,它能计算数字向量的均值:

```
> mean(c(36000, 44000, 56000))
[1] 45333.33
```

这组人的平均收入是 \$45 333.33。从概念上来说,你可以想象这个值是,如果所有收入平等分给每一个人时,每个人应该得到的收入。

回忆先前 `summary()` 函数的输出,它列出了变量 price 和 mileage 的平均值。price 的平均值为 12 962, mileage 的平均值为 44 261,这表明数据集中具有代表性的二手车的价格应该标为 \$12 962,里程表的读数为 44 261。这些告诉我们数据的什么信息呢?因为平均

价格相对偏低，所以可以预期我们数据中包括经济型汽车。当然，数据中也有可能包括新型的豪华汽车，有着高里程数，但是相对较低的平均里程数的统计数据并不提供支持这个假设的证据。另一方面，数据并没有提供证据让我们忽略这个可能性。所以，在进一步检验数据时我们要留意这一点。

尽管到目前为止，均值是最普遍引用的测量数据集中心的统计量，但它不一定是合适的。另一个普遍使用检验中心趋势的指标是**中位数**，它位于排序后值列表的中间。和均值一样，R 提供了函数 `median()` 来获得这个值，可以把它应用到工资数据中，如下所示：

```
> median(c(36000, 44000, 56000))
[1] 44000
```

因为中间值是 44 000，所以收入的中位数是 \$44 000。



如果数据集有偶数个数值，就没有最中间值。在这种情况下，一般是计算按顺序排列的数值列表最中间的两个数值的平均值作为中位数。例如，1、2、3、4 的中位数是 2.5

乍一看，好像中位数和均值是很类似的度量。肯定的是，均值 \$45 333.33 和中位数 \$44 000 并没有太大区别。为什么会有这两种中心趋势呢？这是由于落在值域两端的值对均值和中位数的影响是不同的。尤其是均值，它对**异常值**，或者那些对大多数数据而言异常高或低的值，是非常敏感的。因为均值对异常值是非常敏感的，所以它很容易受到那一小部分极端值的影响而改变大小。

再回忆 `summary()` 函数输出的二手车数据集的中位数。尽管 `price` 的均值和中位数非常相似（相差大约 5%），但 `mileage` 的均值和中位数的不同就非常大了。对于 `mileage` 来说，均值 44 261 比中位数 36 385 大了超过 20%。因为均值比中位数对极端值更敏感，所以均值比中位数大很多这个事实，令我们怀疑数据集中的一些二手车有极高的 `mileage` 值。为了进一步调查这一点，我们需要在分析中应用一些额外的汇总统计量。

2. 测量数据分散程度——四分位数和五数汇总

测量数据的均值和中位数给我们一个迅速概括数据的方法，但是这些中心测度在数值的大小是否具有多样性方面给我们提供了很少的信息。为了测量这种多样性，我们需要应用另一种汇总统计量，它们是与数据的**分散程度**相关的，或者说它们是与数据之间“空隙”的紧密或者松弛有关系的。知道了数据之间的差异，就对数据的最大值和最小值有了了解，同时也会对大多值是否接近均值和中位数有了解。

五数汇总是一组 5 个统计量，它们大致地描述一个数据集的差异。所有的 5 个统计量包含在函数 `summary()` 的输出结果中。按顺序排列，它们是：

- ❑ 最小值 (Min.)。
- ❑ 第一四分位数，或者 Q1 (1st Qu.)。
- ❑ 中位数，或 Q2 (Median)。

□ 第三四分位数，或 Q3 (3rd Qu.)。

□ 最大值 (Max.)。

就像你预期的那样，最小值和最大值是数据集里能发现的最极端的两个值，分别表示数据的最小值和最大值。R 提供了函数 `min()` 和函数 `max()` 来分别计算数据向量中的最小值和最大值。

最小值和最大值的差值称为**值域**。在 R 中，`range()` 函数同时返回最小值和最大值。把 `range()` 函数和差值函数 `diff()` 相结合，这样你能够用一条命令来检验数据的值域：

```
> range(usedcars$price)
[1] 3800 21992
> diff(range(usedcars$price))
[1] 18192
```

第一四分位数和第三四分位数（即 Q1 和 Q3）指的是有 1/4 的值小于 Q1 和有 1/4 的值大于 Q3。它们和中位数（Q2）一起，3 个四分位数把一个数据集分成 4 部分，每一部分都有相同数量的值。



四分位数是**分位数**的一种特殊类型，分位数把数据分为相等数量的数值。除了四分位数外，普遍使用的分位数包括三分位数（分成 3 部分）、五分位数（分成 5 部分）、十分位数（分成 10 部分）和百分位数（分成 100 部分）。百分位数通常用来给数据进行等级评定。例如，一个学生的考试成绩排列在百分位数的第 99 分位数，说明他表现得比其他 99% 的测试者都要好。

我们对 Q1 和 Q3 之间的 50% 的数据特别感兴趣，因为它们就是数据分散程度的一个测度。Q1 和 Q3 之间的差称为**四分位距**（Inter Quartile Range, IQR），可以用函数 `IQR()` 来计算，例如：

```
> IQR(usedcars$price)
[1] 3909.5
```

我们也能从 `summary()` 输出的 `usedcars$price` 变量的结果来手工计算这个值，即计算 $14\ 904 - 10\ 995 = 3909$ 。我们计算的值与 `IQR()` 输出结果之间有差别，这是因为 R 自动对 `summary()` 输出结果进行四舍五入。

`quantile()` 函数提供了稳健的工具来给出一组值的分位数。默认情况下，`quantile()` 函数返回五数汇总的值。把这个函数应用到二手车数据中，将会产生和前面一样的统计量：

```
> quantile(usedcars$price)
 0%      25%      50%      75%     100%
3800.0 10995.0 13591.5 14904.5 21992.0
```



当计算分位数时，有很多种方法处理并排数值和处理没有中间值的数据集。通过指定类型参数，`quantile()` 函数能够在 9 个不同算法之间选择计算分位数的方法。如果你的项目要求一个明确精度的分位数，使用 `?quantile` 命令来阅读该函数的帮助文件是很重要的。

如果我们指定另一个 `probs` 参数，它用一个向量来表示分割点，我们能得到任意的分位数，比如第 1 和第 99 的百分位数可以按如下方式求得：

```
> quantile(usedcars$price, probs = c(0.01, 0.99))
      1%      99%
5428.69 20505.00
```

序列函数 `seq()` 用来产生间距大小相同的向量。这个函数使得获得其他分位数变得很容易，比如要想输出五分位数（5 个组），如下所示：

```
> quantile(usedcars$price, seq(from = 0, to = 1, by = 0.20))
      0%      20%      40%      60%      80%     100%
3800.0 10759.4 12993.8 13992.0 14999.0 21992.0
```

由于理解了五数汇总，所以我们重新检查对二手车数据应用函数 `summary()` 的输出结果。对变量 `price`，最小值是 \$3800、最大值是 \$21 992。有趣的是，最小值和 Q1 之间的差是 \$7000，和 Q3 与最大值的差一样。然而，Q1 和中位数的差，以及中位数和 Q3 的差大约是 \$2000。这就表明上、下 25% 的值的分布比中间 50% 的值更分散，似乎中心周围的值聚集得更加紧密。没有意外的是，我们从变量 `mileage` 也看到了相似的趋势。你在本章后面将会学到，这个差异的模式非常普遍，此时称数据为正态分布。

`mileage` 变量的分散程度同时也表现了另一个有趣的性质：Q3 和最大值之间的差异远大于最小值和 Q1 之间的差异。换句话说，较大的值比较小的值更加分散。

这个发现解释了均值远大于中位数的原因。当中位数相对在同一位置时，因为均值对极端值更敏感，所以均值会被极端大的值拉高。这是一个很重要的性质，当数据直观呈现出来时就更显而易见。

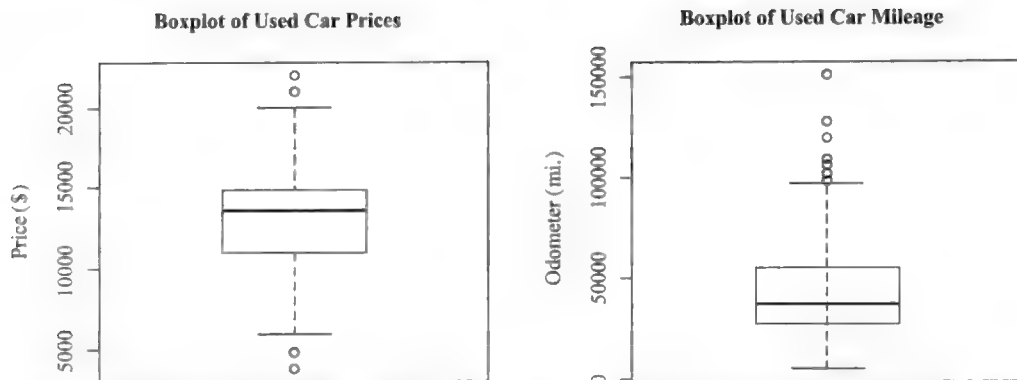
3. 数值型变量可视化——箱图

可视化数值型变量对数据中很多问题的诊断都是有幫助的。一种对五数汇总的常用可视化方式是箱图。箱图以一种特定方式显示数值型变量的中心和分散程度，这种方式使你能很快了解变量的值域和偏度，或者它还可以和其他变量做比较。

下面观察二手车数据的变量 `price` 和变量 `mileage` 的箱图。要想得到一个变量的箱图，可以使用函数 `boxplot()`。我们也将指定一些其他参数——`main` 和 `ylab`，它们分别为图形加一个标题和标注图形的 y 轴（即竖轴）。创建变量 `price` 和变量 `mileage` 箱图的命令是：

```
> boxplot(usedcars$price, main="Boxplot of Used Car Prices",
          ylab="Price ($)")
> boxplot(usedcars$mileage, main="Boxplot of Used Car Mileage",
          ylab="Odometer (mi.)")
```

R 将产生下面的图形：



箱图用水平线来表示五数汇总的值。每个图中，构成每个盒子的中间水平线从下向上，依次代表 Q1、Q2（中位数）和 Q3。中位数用粗黑线表示，对变量 `price` 而言，这条线在竖轴上的纵坐标是 \$13 592；对变量 `mileage`，这条线的纵坐标是 36 385。



像上面图形所示的简单箱图中，箱图的宽度是随意的，它不能说明任何数据的特征。为了满足更加复杂分析的需要，通过用盒子的形状和尺寸，对多组数据进行比较是可能的。要想知道更多关于箱图的这种特征的信息，可以通过 `?boxplot` 命令，查询 R 中 `boxplot()` 文件的帮助文件的 `notch` 和 `varwidth` 选项。

最小值和最大值是用细线（whisker）来表示的，就是在盒子下面和上面的细线。然而，通常仅允许细线延伸到最小为低于 Q1 的 1.5 倍 IQR 的最小值，或者延伸到最大为高于 Q3 的 1.5 倍 IQR 的最大值。任何超出这个临界值的值都认为是异常值，并且用圆圈或者点来表示。例如，变量 `price` 的 IQR 是 3909，Q1 是 10 995，Q3 是 14 904。因此异常值是任何小于 $10\,995 - 1.5 \times 3909 = 5131.5$ 或者大于 $14\,904 + 1.5 \times 3909 = 20\,767.5$ 的值。

箱图在高端和低端都会出现这类异常值。在 `mileage` 的箱图中，在低端没有这样的异常值，所以底部的细线延伸到最小值 4867。在高端，我们看到了几个比 100 000 英里大的异常值。那些异常值就可以解释我们前面探索中所发现的问题，即解释了均值远大于中位数。

4. 数值型变量可视化——直方图

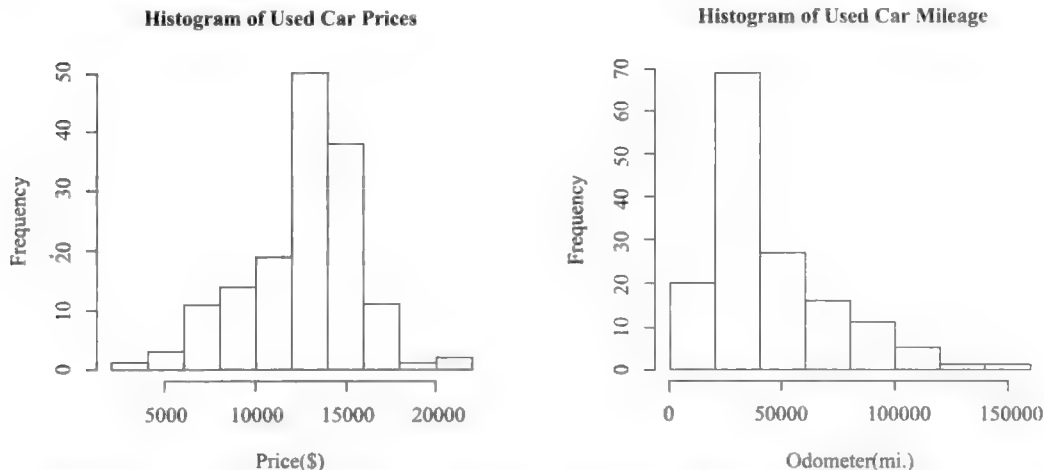
直方图（histogram）是另一种形象化描述数值型变量间差异的方式。它和箱图相似的地方在于，它也把变量值按照预先设定的份数进行分隔，或者说按照预先定义的容纳变量值的分段进行分隔。箱图把数据分成 4 部分，且每部分必须包含相同数量的值，根据需要分段也可以变宽或变窄。相反，直方图可以有相同宽度的任意数量的分段，但是分段可以包含不同数量的值。

可以用函数 `hist()` 为二手车数据的变量 `price` 和 `mileage` 绘制直方图。就像我们绘制箱图时那样，可以用参数 `main` 来指定图形的标题，用参数 `xlab` 给出标记 `x` 轴。绘制

直方图的命令如下：

```
> hist(usedcars$price, main = "Histogram of Used Car Prices",
      xlab = "Price ($)")
> hist(usedcars$mileage, main = "Histogram of Used Car Mileage",
      xlab = "Odometer (mi.)")
```

产生的直方图如下图所示。



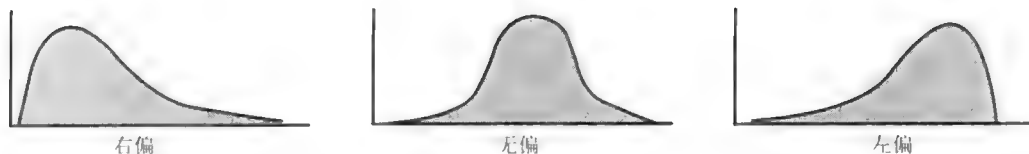
直方图是由一系列的竖条组成，其高度表示落在等长的划分数据值的分段内的数据值的个数或频率。分割每一个竖条的垂线，就像横坐标的标签一样，表明分段内值的起始点和终点。

例如，在变量 `price` 的直方图上，10 个竖条中的每一个都表示范围为 \$2000 的分段，这些分段的范围是从 \$2000 开始，到 \$14 000 结束。直方图中间最高的竖条代表的分段范围为 \$12 000 ~ \$14 000，频率是 50。因为我们知道数据中有 150 辆汽车，其中 1/3 汽车的价格是在 \$12 000 ~ \$14 000 之间。接近 90 辆汽车（超过一半），报价在 \$12 000 ~ \$16 000 之间。

变量 `mileage` 的直方图包括 8 个竖条，它表明每个分段长度都是 20 000 英里，值域从 0 开始，到 160 000 英里结束。与变量 `price` 的直方图不一样的是，在变量 `mileage` 的直方图中最高的竖条不在数据的中心，而是在直方图的左侧。这个最高竖条所在的分段中有 70 辆车，里程表的范围从 20 000 ~ 40 000 英里。

你可能也注意到了两个直方图的形状有一点不同。似乎二手车 `price` 的图形趋向于平均分布在中心的两侧，而汽车 `mileage` 的图形则偏到了左侧。这个性质称为**偏度**（*skew*），具体来说说是右偏，因为高端的值（右侧）和低端的价值（左侧）相比更加分散。如下图所示，偏斜数据的直方图看上去偏到了一边。

能在数据中快速诊断出这类模式是直方图作为数据探索工具的优点之一。在我们检验其他数值型数据模型的模式时，这个优点将更为重要。



5. 了解数值型数据——均匀分布和正态分布

描述数据的中心和分散程度的直方图、箱图和统计量都提供了检验变量分布的方法。变量的分布描述了一个值落在不同值域里的可能性大小。

如果所有值都是等可能发生，这个分布就称为均匀分布，例如，记录投掷一个均匀的六边形骰子所得结果的数据集。容易用直方图来探测出一个均匀分布，因为其直方图的竖条大致有一样的高度。当用直方图来可视化数据时，它可能如左下图所示。

要注意的重要一点是，并非所有的随机事件都服从均匀分布。例如，掷一个六边重量不同的魔术骰子，将使得一些数字发生的概率比其他的大。每一次掷骰子会产生一个随机数字，但 6 个数字出现的概率不相等。

例如，下面回到前面的二手车数据。很明显这个数据不是均匀分布的，因为有些值明显比其他值发生的可能性更大。事实上，在变量 `price` 的直方图上，可以看出中心值两边的值，偏离中心越远，发生的频率就越小，这就是一个钟形的数据分布。这个特征在现实世界的数据中非常普遍，它成为所谓的正态分布的标志性特征。钟形曲线的典型形状如右下图所示。



尽管有许多非正态分布的类型，但许多现象产生的数据都可以用正态分布来描述。因此，正态分布的性质已被研究得很透彻了。

6. 衡量数据的分散程度——方差和标准差

分布使我们能够用少量的参数来描述大量值的特性。用来描述现实生活中大量数据的正态分布，可以用两个参数来定义：中心和分散程度。正态分布的中心可以用均值来定义，正如我们在前面使用的那样。分散程度通过一种称为标准差的统计量来测量。

为了计算标准差，我们必须先获得方差，它定义为每一个值与均值之间的平方差的均值。用数学符号表示，一组具有 n 个值的变量 x 的方差可以通过下面的公式定义。希腊字母 μ 表示值的均值，方差用希腊字母 σ 的平方来表示：

$$\text{Var}(X) = \sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

标准差就是方差的平方根，用 σ 来表示，如下所示：

$$\text{StdDev}(X) = \sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$$

要想在 R 中获得方差和标准差，可以应用函数 `var()` 和函数 `sd()`。例如，计算变量 `price` 与变量 `mileage` 的方差与标准差，如下所示：

```
> var(usedcars$price)
[1] 9749892
> sd(usedcars$price)
[1] 3122.482
> var(usedcars$mileage)
[1] 728033954
> sd(usedcars$mileage)
[1] 26982.1
```

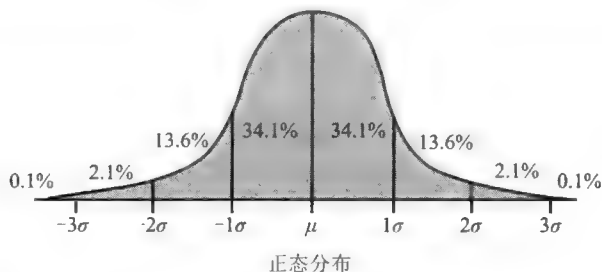
当我们解释方差的时候，方差越大表示数据在均值周围越分散。标准差表示平均来看每个值和均值相差多少。



如果你用上面的公式手工来计算这些统计量，你得出的结果将会与 R 的内置函数得出的结果略有不同。这是因为上面的公式给出的是总体方差（除以 n ），而 R 内置函数用的是样本方差（除以 $n-1$ ）。除非数据集很小，否则这两种结果的区别是很小的。

在假设数据服从正态分布的条件下，标准差能用来快速地估计出一个给定值有多大程度的偏大或者偏小。68-95-99.7 规则说明正态分布中 68% 的值落在均值左右 1 个标准差的范围内，而 95% 和 99.7% 的值各自落在均值左右 2 个和 3 个标准差的范围内。这个规则可以由下图来说明。

把这个知识应用到二手车数据中，我们知道变量 `price` 的均值是 \$12 962，所以该数据中大约有 68% 的车的价格在 \$9840 ~ \$16804 之间。尽管 68-95-99.7 规则仅仅局限于正态分布中，但是这个基本准则能应用到所有的数据中，数值落在均值的 3 个标准差以外是极端罕见的事件。



2.5.3 探索分类变量

如果你回忆一下，你会发现二手车数据集有 3 个分类变量：`model`、`color` 和 `transmission`。

因为在加载数据时，我们用到了 `stringsAsFactors = FALSE` 参数，所以 R 把它们作为字符型 (`chr`) 变量而不是自动把它们转化成因子。此外，我们可能考虑把 `year` 看作分类变量。尽管它是数值型 (`int`) 的，但是每一个 `year` 值是一个类别，该类别可以应用到多辆汽车上。

与数值型数据相比，分类数据是用表格而不是汇总统计量来进行检测的。表示单个分类变量的表格称为一元表。函数 `table()` 能用来产生二手车数据的一元表：

```
> table(usedcars$year)
2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
   3    1    1    1    3    2    6   11   14   42   49   16    1
> table(usedcars$model)
SE SEL SES
78  23  49
> table(usedcars$color)
Black  Blue  Gold  Gray  Green  Red Silver White Yellow
   35    17     1   16     5   25   32    16     3
```

输出的表格结果列出了名义变量的不同类别和该类别的值的数量。由于我们知道数据集一共有 150 个二手车数据，所以我们能确定其中大约有 1/3 是在 2010 年制造的，因为 $49/150$ 大约是 33%。

R 同时也能在 `table()` 函数产生的表格上直接应用函数 `prop.table()`，计算表格中格子的比例，如下所示：

```
> model_table <- table(usedcars$model)
> prop.table(model_table)
      SE      SEL      SES
0.5200000 0.1533333 0.3266667
```

型号为 SE 的汽车的比例是 0.520 000 0，所以有 52% 的汽车是 SE 这种车型。

函数 `prop.table()` 的结果能与其他 R 函数相结合来转换输出的结果。假设我们想要把结果用保留一位小数的百分数来表示，就可以把各个比例值乘以 100，再用 `round()` 函数并且指定 `digits = 1` 来实现，如下所示：

```
> color_table <- table(usedcars$color)
> color_pct <- prop.table(color_table) * 100
> round(color_pct, digits = 1)
Black  Blue  Gold  Gray  Green  Red Silver White Yellow
 23.3   11.3   0.7  10.7   3.3  16.7  21.3  10.7   2.0
```

尽管它包含的信息和 `prop.table()` 函数默认的输出结果一样，但是相对来说这样更容易阅读。结果显示 `black` 是最普遍的颜色，因为广告列出的所有汽车中将近有 1/4 (23.3%) 是 `black` 的。`silver` 与之接近，排在第二位，有 21.3%；`red` 是第三，有 16.7%。

衡量中心趋势——众数

在统计术语中，一个特征（即变量）的众数是指出现最频繁的那个值。与均值和中位数一样，众数是另一个测量中心趋势的统计量。它通常应用在分类数据中，因为均值和中位数

并不是为名义变量定义的。

例如，在二手车数据中，`year` 变量的众数是 2010，而 `model` 和 `color` 的众数分别为 SE 和 Black。一个变量可能有多个众数；只有一个众数的变量是**单峰的**，有两个众数的变量为**双峰的**。有多个众数的数据通常称为**多峰的**。



尽管你可能猜测能用 `mode()` 函数得到众数，但是 R 却是用这个函数来得出变量的类型（如数值型，列表等），而不是统计量众数。相反，为了找到统计量众数，只需要查看表格输出结果中具有最大值的类别即可。

众数是从定性角度来了解数据集中的重要值。然而，太关注众数有可能很危险，因为最常见的值并不一定就是绝大多数。例如，尽管 Black 是二手车变量 `color` 的众数，但是 Black 仅占有所有列出汽车的 1/4。

考虑众数时最好把它和其他的类别联系起来：是否有一个类别占主导地位，或者多个类别占主导地位？据此，我们可能会问：最常见的值告诉我们被测量变量的哪些信息。如果 Black 和 Silver 是最普遍使用的二手车颜色，那么我们可以假设数据是从奢华汽车中得来的，这类汽车趋向于销售更加保守的颜色；或者它们也可能是经济型汽车，这类汽车有更少可供选择的颜色。在我们进一步检验这些数据时，我们要记住这些问题。

把众数考虑成最普遍的值，这使得我们能够把统计量众数的概念应用到数值型数据。严格地说，连续变量是不可能有多数的，因为没有两个值可能是重复的。然而，如果我们把众数考虑成直方图中最高的那个竖条，就能够讨论如变量 `price` 和变量 `mileage` 的众数。当探索数值型数据时，考虑众数是很有帮助的，特别要检验数据是否为多峰的。

2.5.4 探索变量之间的关系

到目前为止，我们一次只检验一个变量，只计算**单变量**统计量。在我们的研究过程中，我们列举了当时尚不能回答的一些问题：

□ `price` 数据有没有暗示我们只检验了经济类的汽车，还是检验的数据里也包括高里程的奢华汽车呢？

□ `model` 和 `color` 数据之间的关系，有没有提供关于我们所检验的汽车类型的洞察呢？

这类问题能通过关注**二变量**关系，即考虑两个变量之间的关系来进行处理。超过两个变量间的关系称为**多变量**关系。下面从二变量的情况开始讨论。

1. 变量之间关系的可视化——散点图

散点图是一种可视化二变量之间关系的图形。它是一个两维的图形，将点画在坐标平面中，该坐标平面的横坐标 x 是其中一个特征的值，纵坐标 y 由另一个特征的值来标识。坐标平面上点的排放模式，揭示了两个特征之间的内在关系。

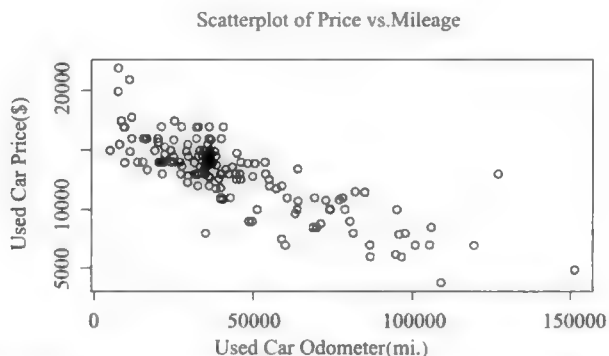
为了回答变量 `price` 和 `mileage` 之间的关系，下面来分析一个散点图。我们将用 `plot()` 函数以及在前面绘图中用过的标记图形的参数 `main`、`xlab` 和 `ylab`。

为了使用 `plot()` 函数，我们需要指定 `x` 向量和 `y` 向量，它们含有给图形中的点定位的值。尽管无论用哪个变量来表示 `x` 坐标和 `y` 坐标，结论都是一样的，但是惯例规定，`y` 变量是假定依赖于另一个变量的变量（因此称为**因变量**）。因为里程表的读数不能被卖家修改，所以它不可能由汽车的价格决定。相反，我们假设 `price` 是由里程表的里程数（`mileage`）决定的。因此，我们将把 `price` 作为 `y`，或者称**因变量**。

绘制散点图的全部命令如下：

```
> plot(x = usedcars$mileage, y = usedcars$price,
      main = "Scatterplot of Price vs. Mileage",
      xlab = "Used Car Odometer (mi.)",
      ylab = "Used Car Price ($)")
```

这会产生下面的散点图：



使用散点图，我们可以了解二手车的价格和里程表的读数之间的一个清晰的关系。为了研究这张图，我们观察当 `x` 轴变量的值增加时，`y` 轴变量的值是如何改变的。在这个例子中，当 `mileage` 值逐渐增加时，`price` 值变得越来越低，它表明随着 `mileage` 的增加，广告中列出的价格会降低。如果你曾经卖过或者买过二手车，这一点不难得到。

一个更有趣的发现可能是，除了 125 000 英里和 \$14 000 所构成的一个异常点以外，有很少一部分汽车同时有很高的 `price` 和很高的 `mileage`。缺少更多这样的点，就提供了证据来支持下列结论：数据中不可能包含高里程的奢华汽车。数据中所有贵的汽车，特别是那些 \$17 500 以上的汽车，看上去都有超低的里程数，这暗示我们可能看到的是一类全新的卖价为 \$20 000 的汽车。

变量 `price` 和变量 `mileage` 之间的关系是负相关的，因为散点图是一条向下倾斜的直线。正相关看起来是形成一条向上倾斜的直线。一条水平的线，或者一个看上去随机分布的点集，证明两个变量完全不相关。两个变量之间线性关系的强弱是通过统计量**相关系数**来测量的。相关系数在第 6 章中详细讨论，第 6 章将学习如何使用回归方法来建立线性关系。



注意不是所有的关联都形成直线。有时点会形成一个 U 形或者 V 形；有时关联模式看上去随着 `x` 变量或者 `y` 变量的增加而变弱。这样的模式说明两个变量之间的关系不是线性的。

2. 检验变量之间的关系——双向交叉表

为了检验两个名义变量之间的关系，使用**双向交叉表** (two-way cross-tabulation，也称为**交叉表**或者**列联表**)。交叉表和散点图相类似，它允许你观察一个变量的值是如何随着另一个值的变化而变化的。双向交叉表的格式是：行是一个变量的水平，列是另一个变量水平。每个表格的单元格中的值用来表明落在特定行、列的单元格中值的数量。

为了回答我们关于 model 和 color 之间关系的问题，我们观察一个交叉表。R 中的多个函数都能生成双向表，包括 table() 函数，我们也可以把 table() 函数用在单向表中。由 Gregory R. Warnes 创建的 gmodels 添加包里的 CrossTable() 函数可能是用户最喜欢用的函数，因为它在一个表格中出现了行、列和边际百分比，省去了我们要自己组合这些数据的麻烦。要想安装 gmodels 添加包，输入：

```
> install.packages("gmodels")
```

在安装了添加包后，仅需要输入命令 library(gmodels) 来加载该添加包。在每次用到 CrossTable() 函数时，你需要在 R 系统中加载这个添加包。

在我们进一步分析以前，让我们通过减少 color 变量中水平的数量来简化我们的任务。这个变量有 9 个水平，但是我们并不是真的需要如此详细。我们真正感兴趣的是汽车的颜色是否是保守的。为了这个目的，我们将把 9 种颜色分为两组：第一组将包括保守的颜色：Black、Gray、Silver 和 White；第二组将包括 Blue、Gold、Green、Red 和 Yellow。我们创建一个二元指示变量（常常称为哑变量），根据我们的定义来表示汽车的颜色是否是保守的。如果是保守的颜色，指示变量的值就是 1，否则值为 0。

```
> usedcars$conservative <-  
  usedcars$color %in% c("Black", "Gray", "Silver", "White")
```

这里，你可能注意到一个新的命令：“%in%”操作符，它根据左边的值是否在右边的向量中，为操作符左边向量中的每一个值返回 TRUE 或者是 FALSE。简单地说，你可以理解为“这辆二手车的颜色是在 black、gray、silver 和 white 这组中吗？”

观察 table() 得到的我们新建变量的输出结果，我们看到 2/3 的汽车有保守的颜色，而 1/3 的汽车没有保守的颜色：

```
> table(usedcars$conservative)  
FALSE  TRUE  
   51    99
```

现在，让我们看看交叉表中 conservative (保守) 颜色汽车的比例是如何随着 model 变化而变化的。因为我们假设汽车的型号决定了颜色的选择，所以我们把 conservative 作为因变量 (y)。CrossTable() 命令的应用如下：

```
> CrossTable(x = usedcars$model, y = usedcars$conservative)
```

由此产生了下面的表格：

Cell Contents	
Chi-square contribution	N
N / Row Total	
N / Col Total	
N / Table Total	

Total Observations in Table: 150

usedcars\$model	usedcars\$conservative		Row Total
	FALSE	TRUE	
SE	27	51	78
	0.009	0.004	0.520
	0.346	0.654	
	0.529	0.515	
	0.180	0.340	
SEL	7	16	23
	0.086	0.044	0.153
	0.304	0.696	
	0.137	0.162	
	0.047	0.107	
SES	17	32	49
	0.007	0.004	0.327
	0.347	0.653	
	0.333	0.323	
	0.113	0.213	
Column Total	51	99	150
	0.340	0.660	

CrossTable() 的输出中包含了大量数据。最上面的一张图（标示为 Cell Contents）说明如何解释每一个值。表格的行表示了二手车的 3 个型号：SE、SEL 和 SES（再加上额外的一行用来表示所有型号的汇总）。表格的列表示汽车的颜色是否是保守的（加上额外的一列对所有两种颜色求和）。每个格子中的第一个值表示那个型号和那个颜色的汽车的数量。比例分别表示这个格子的卡方统计量，以及在行、列和整个表格中占的比例。

在表格中，我们最感兴趣的是保守颜色汽车占每一种型号的行比例。行比例告诉我们 0.654（65%）的 SE 汽车用保守的颜色，SEL 汽车的这个比例是 0.696（70%），SES 汽车是 0.653（65%）。这些数值的差异相对来说是较小的，这暗示不同型号的汽车选择的颜色类型没有显著的差异。

卡方值指出了在两个变量中每个单元格在皮尔森卡方独立性检验中的贡献。这个检验测量了表格中每个单元格内数量的不同只是由于偶然的可能性有多大。如果概率值是非常低的，那么就提供了充足的证据表明两个变量是相关的。

你能在引用 CrossTable() 函数时增加一个额外的参数，指定 `chisq = TRUE` 来获得卡方检验的结果。在我们的案例中，概率值是 93%，暗示格子内数量的变化很可能仅仅是由于偶然，而不是在 `model` 和 `color` 之间真的存在关联。

2.6 总结

在本章中，我们学习了在 R 中管理数据的基础。从深入剖析用来存储不同类型数据的数

据结构开始。R 数据的基本结构是向量，它扩展和组合成更加复杂的数据结构，比如，列表和数据框。数据框是与数据集概念相联系的 R 数据结构，数据框内同时有特征和案例。

我们同时也学习了如何从不同的数据源把数据导入 R 中。R 提供了从 CSV 文件提取数据和把数据存储于 CSV 文件的函数。SQL 数据集能用 RODBC 添加包查询。

最后，我们将这些技能应用于包含二手车价格的真实数据集中。我们用常用的中心趋势和分散程度统计量来检验数值型变量，用散点图来可视化价格和里程表读数。我们用表格检验名义变量。在检验二手车数据时，我们采用这种可以用来探索所有数据集的分析过程。

既然我们花了些时间来理解 R 中数据管理的基础，就已经准备好了使用机器学习来解决真实世界的问题。第三章，我们将用最近邻方法着手处理我们的第一个分类任务。

懒惰学习——使用近邻分类

最近，我阅读了一篇描述一种新型用餐体验的文章，顾客在一个完全黑暗的餐厅里接受服务，而服务员在仅凭触觉和听觉记忆的路上小心地移动。这些餐厅的魅力植根于这样的思想：去掉一个人的视觉感官输入将会增强他的味觉和嗅觉，从而可以使他以一种全新并且感到兴奋的方式来体验食物。用餐者的每一口都认为是一个小小的冒险，在这一个小小的冒险中，他们将会发现厨师所准备的美味。

你能够想象用餐者是如何体验看不到的食物吗？刚开始可能会有一个数据收集的短暂阶段：突出的香料、香味和口感是什么？食物尝起来是咸还是甜？利用这些数据，顾客接下来可能会将这一小口的食物与他之前的体验进行对比，海水的味道可能会引起海鲜的印象，而泥土的味道可能会与过去包含菌类植物的饭菜联系在一起。就个人而言，我用一句稍加修改的谚语来想象这一探索过程：如果该食物闻起来像只鸭子，并且尝起来也像只鸭子，那么你就很可能在吃鸭子。

这阐述了一个可以用于机器学习思想——就像另一句有关鸟类的格言：有一样羽毛的鸟会聚集在一起（即“物以类聚，人以群分”）。换句话说就是：相似的东西很可能具有相似的属性。利用这个原理，我们可以对数据进行分类，将其划分到最相近的类别或者最接近的邻居。本章将专门讨论使用这种方法进行分类，你将会学到：

- 定义近邻分类器的关键概念，以及为什么它们被认为是“懒惰”(lazy)学习器。
- 通过距离来测量两个案例相似性的方法。
- 如何使用 kNN (k-Nearest Neighbors, **k 近邻**) 算法的 R 语言实现来诊断乳腺癌。

如果所有这些关于食物的话题让你感到饿了，那么你可能想要吃些点心了。我们的第一个任务就是通过对 kNN 方法的使用和安排一个持续较长有关烹饪的讨论来理解 kNN 方法。

3.1 理解使用近邻进行分类

用一句话来说，近邻分类器就是把未标记的案例归类为与它们最相似的带有标记的案例所在的类。尽管这个想法很简单，但是近邻分类方法是极其强大的，它们已经成功地应用在下列领域中：

- 计算机视觉应用，包括在静止图像和视频中的光学字符识别和面部识别。
- 预测一个人是否喜欢推荐给他们的电影（就像 Nextfix 公司的比赛）。
- 识别基因数据的模式，用于发现特定的蛋白质或者疾病

一般来说，近邻分类器非常适用于这样的分类任务，其中的特征和目标类之间的关系众多、复杂，用其他方式极难理解，但是具有相似类的项目又是非常近似。换个说法，也就是说，如果一个概念很难定义，但是当你看到它时你知道它是什么，那么近邻分类就可能是合适的方法。另一方面，如果组与组之间没有明确的界限，那么该算法总的来说不太适合用来确定类边界。

3.1.1 kNN 算法

用于分类的近邻方法是通过 kNN 算法实现的，让我们来看一下该算法的优缺点：

优点	缺点
<ul style="list-style-type: none"> ● 简单且有效 ● 对数据的分布没有要求 ● 训练阶段很快 	<ul style="list-style-type: none"> ● 不产生模型，在发现特征之间关系上的能力有限 ● 分类阶段很慢 ● 需要大量的内存 ● 名义变量（特征）和缺失数据需要额外处理

kNN 算法开始于一个分成几个类别的案例所组成的训练数据集，类别由名义变量来标记。假设我们有一个由未标记的案例构成的测试数据集，除去分类标记外，测试数据集和训练数据集有相同的特征。对于测试数据集中的每一个记录，kNN 确定训练数据集中与该记录相似度“最近”的 k 条记录，其中 k 是一个预先指定的整数，未标记的测试实例被分配到 k 个近邻中占比最大的那个类中。

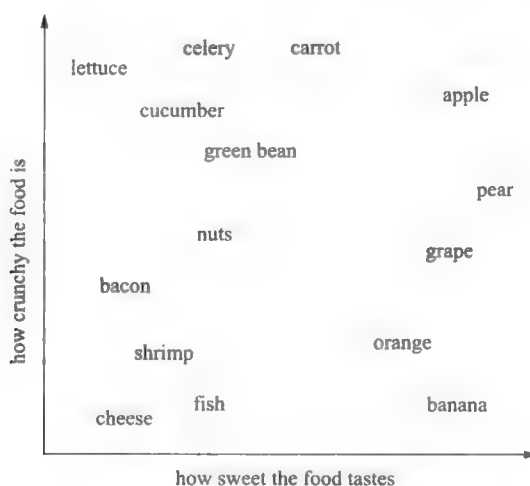
为了说明这个过程，让我们回顾引言中所描述的黑暗餐厅用餐的经历。假设在吃一顿神秘的膳食之前，我们创建了一个品味数据集，在这个数据集中，记录了我们对于之前所品尝的很多配料的印象。为了简单起见，我们只记录了每种配料（ingredient）的两个特征，第一个特征是对配料有多脆的度量（crunchiness），从 1 ~ 10；第二个特征是对配料有多甜的度量（sweetness），从 1 ~ 10。然后，我们标记配料为 3 种类型之一：fruit（水果）、vegetable（蔬菜）或者 protein（蛋白质）。该数据集的前几行可能具有如下所示的结构：

ingredient	sweetness	crunchiness	food
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit

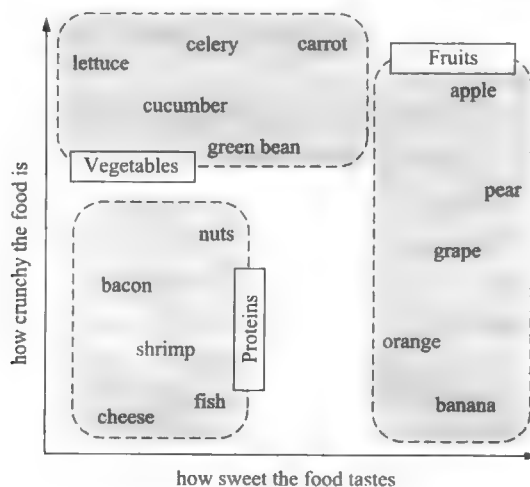
(续)

ingredient	sweetness	crunchiness	food
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

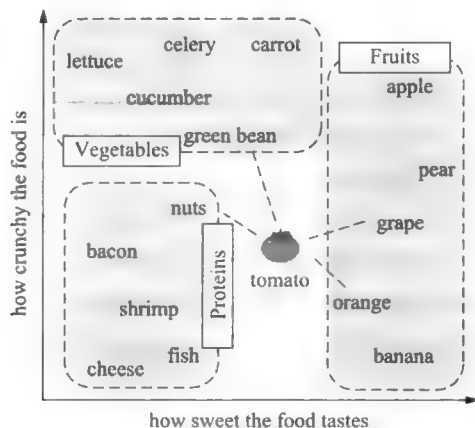
kNN 算法将特征处理为一个多维**特征空间**（feature space）内的坐标。由于我们的数据集只包含了两个特征，所以特征空间是二维的。我们可以绘制二维数据散点图，维度 x 表示配料的甜度（sweetness），维度 y 表示配料的脆度（crunchiness）。在品味数据集中增加更多的配料后，散点图看起来可能如下图所示。



你注意到上述图形了吗？相似类型的食物趋向于聚集得更近。如下图所示，蔬菜往往是脆而不甜的；水果往往是甜的，有可能脆，也有可能不脆；而蛋白质往往是既不脆也不甜。



假设在创建此数据集之后，我们决定用它来解决一个古老的问题：西红柿（tomato）是水果（fruit），还是蔬菜（vegetable）？我们可以使用一种近邻方法来确定哪类更适合西红柿（tomato），如下图所示。



1. 计算距离

定位西红柿（tomato）的近邻需要一个**距离函数**（distance function）或者一个用来衡量两个实例（即案例）之间相似性的公式。

计算距离有许多种不同的方法。传统上，kNN算法采用的是**欧式距离**（Euclidean distance），这种距离可以通过用尺子连接两个点来测量，在上图中，我们通过虚线将西红柿（tomato）与它的邻居连接在一起。



欧式距离通过“直线距离”（crow flies）来度量，即最短的直接路线。另一种常见的距离度量是**曼哈顿距离**（Manhattan distance），该距离基于一个行人在城市街区步行所采取的路线。如果你有兴趣了解更多关于距离度量的方法，可以使用 `?dist` 命令，阅读 R 中的距离函数文档（该文档本身就是一个很有用的工具）。

欧式距离通过如下公式定义，其中 p 和 q 是需要比较的案例，它们都有 n 个特征，项 p_1 代表案例 p 的第一个特征的值，而 q_1 代表案例 q 的第一个特征的值：

$$\text{dist}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

距离公式涉及比较每个特征的值。例如，为了计算西红柿（tomato）（sweetness（甜度）= 6, crunchiness（脆度）= 4）和绿豆（green bean）（sweetness（甜度）= 3, crunchiness（脆度）= 7）之间的距离，可以使用如下的公式：

$$\text{dist}(\text{tomato}, \text{green bean}) = \sqrt{(6-3)^2 + (4-7)^2} = 4.2$$

与此类似，可以计算西红柿 (tomato) 与它的几个近邻之间的距离，如下表所示。

ingredient	sweetness	crunchiness	food type	distance to the tomato
grape	8	5	fruit	$\text{sqrt}((6-8)^2 + (4-5)^2) = 2.2$
green bean	3	7	vegetable	$\text{sqrt}((6-3)^2 + (4-7)^2) = 4.2$
nuts	3	6	protein	$\text{sqrt}((6-3)^2 + (4-6)^2) = 3.6$
orange	7	3	fruit	$\text{sqrt}((6-7)^2 + (4-3)^2) = 1.4$

为了将西红柿 (tomato) 归类为蔬菜 (vegetable)、蛋白质 (protein)，或者水果 (fruit)，我们先从把西红柿 (tomato) 归类到离它最近的一种食物所在的类型开始。因为这里 $k=1$ ，所以这称为 1NN 分类。橙子 (orange) 是西红柿 (tomato) 的近邻，距离是 1.4。因为橙子 (orange) 是一种水果，所以 1NN 算法把西红柿 (tomato) 归类为一种水果。

如果我们使用 $k=3$ 的 kNN 算法，那么它会在 3 个近邻：橙子 (orange)、葡萄 (grape) 和坚果 (nuts) 之间进行投票表决。因为这 3 个邻居的大多数归类为水果 (2/3 的票数)，所以西红柿 (tomato) 再次被归类为水果。

2. 选择一个合适的 k

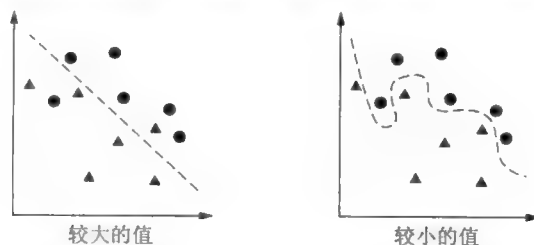
确定用于 kNN 算法的邻居数量将决定把模型推广到未来数据时模型的好坏。过度拟合和低度拟合训练数据之间的平衡问题称为**偏差-方差权衡** (bias-variance tradeoff)。选择一个大的 k 会减少噪声数据对模型的影响或者减少噪声导致的模型波动，但是它会使分类器产生偏差。比如，它有忽视不易察觉但却很重要模式的风险。

假设我们采取一个极端的情况，即设置一个非常大的 k ，它等于训练数据中所有观测值的数量。由于每一个训练案例都会在最后的投票表决中出现，所以最常见的训练类总会获得大多数的票。因此，模型总会预测为数量占大多数的训练类，而不管哪个邻居是最近的。

在相反的极端情况下，使用一个单一的近邻会使得噪声数据或者异常值过度影响案例的分类。例如，假设一些训练案例被意外地贴错了标签，而另一些未标记的案例恰好是最接近于被错误标记的训练案例，那么它就会被预测到错误的类中，即使其他的 9 个近邻有不同的投票。

显然，最好的 k 值应该取这两个极端值之间的某个值。

下面的图更一般地说明了决策边界 (由虚线表示) 如何受到较大的或者较小的 k 值的影响。较小的 k 值会给出更复杂的决策边界，它可以更精细地拟合训练数据。但问题是，我们并不知道是直线边界还是曲线边界能更好地代表我们将要学习的正确概念。



在实际中， k 的选取取决于要学习概念的难度和训练数据中案例的数量。通常， k 为 $3 \sim 10$ 。一种常见的做法就是设置 k 等于训练集中案例数量的平方根。在我们之前研究的食物分类器中，我们可以设置 $k=4$ ，因为训练数据中有 15 个案例，15 的平方根是 3.87。

然而，这样的规则可能并不总是产生一个最好的 k 值。另一种方法是基于各种测试数据集来测试多个 k 值，并选择一个可以提供最好分类性能的 k 值。从另一方面来说，除非数据的噪声非常大，否则更大的、更具代表性的训练数据集可以使 k 值的选择并不那么重要。这是因为即使是微小的概念，也将有一个足够大的案例池来进行投票，以便选举出近邻。



一个不太常见但很有趣的解决这个问题的方法是选择一个较大的 k 值，但是同时应用一个**权重投票**（weighted voting）过程，在这个过程中，认为较近邻的投票比远邻的投票更加有权威。

3. 准备 kNN 算法使用的数据

在应用 kNN 算法之前，通常将特征转换为一个标准的范围内。这个步骤的合理性在于，距离公式依赖于特征是如何被度量的。特别地，如果某个特征具有比其他特征更大的值，那么距离的度量就会强烈地被这个较大的值所支配。而对于我们之前的食物品味数据，这并不算是一个问题，因为甜度（sweetness）和脆度（crunchiness）的度量都是在 $1 \sim 10$ 的范围内。

假设我们增加一个额外的特征来表示辛辣度（spiciness），并使用**史高维尔指标**（Scoville scale）来测量它。史高维尔指标是辛辣热量的一种标准化度量，范围从 0（不辣）到超过 100 万（最火辣的辣椒）。因为辛辣食物和非辛辣食物之间的差异可能会超过 100 万，而甜食和非甜食之间的差异至多是 10，所以我们可能会发现距离度量只能通过食物的辛辣度来加以区别，而脆度和甜度的影响将会由于辛辣度对于距离的贡献而显得相形见绌。

我们需要的是—种“收缩”或者重新缩放各种特征的方法，以使得每个特征对距离公式的贡献相对平均。例如，如果甜度和脆度的度量都在范围 $1 \sim 10$ ，那么我们也希望辛辣度的度量也在范围 $1 \sim 10$ ，有一些方法可以完成这样的尺度调整。

对 kNN 算法的特征进行重新调整的传统方法是**min-max 标准化**（min-max normalization），该过程将特征转化，以使它的所有值都落在 $0 \sim 1$ 的范围内。将特征进行 min-max 标准化的公式如下所示。本质上，该公式就是特征 X 的每一个值减去它的最小值再除以特征 X 的值域：

$$X_{\text{new}} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

min-max 标准化的特征值可以这样解释：按 $0\% \sim 100\%$ 来说，在原始最小值和原始最大值范围内，原始值到原始最小值的距离有多远。

另一种常见的变换称为**z-score 标准化**（z-score standardization）。下面的公式是减去特征 X 的均值后，再除以 X 的标准差：

$$X_{\text{new}} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$

这个公式是基于在第2章中所介绍的正态分布的性质，即根据每一个特征的值落在均值上下的标准差的数量来重新调整每一个特征的值，所得到的值称为 **z-score**。z-score 落在一个无界的负数和正数构成的范围内，它们不像 min-max 标准化后的值那样，它们没有预定义的最小值和最大值。

欧式距离公式并不是为名义数据定义的。因此，为了计算名义特征之间的距离，我们需要将它们转化成数值型格式。一种典型的解决方案就是利用**哑变量编码**（dummy coding），其中 1 表示一个类别，0 表示其他类别。例如，对性别变量进行哑变量编码可以这样构建：

$$\text{male} = \begin{cases} 1 & \text{如果 } x = \text{male} \\ 0 & \text{其他} \end{cases}$$

注意对含有两个可能取值的（二元）性别变量进行哑变量编码如何产生一个新的名为 **male** 的特征，而为 **female** 构建一个单独的特征是没有必要的，因为两种性别是互斥的，知道其中一个就足够了。

这种方法实际上可以更加广泛地应用。一个具有 n 个类别的名义特征可以通过对特征的 $(n-1)$ 个水平创建二元指示变量来进行哑变量编码。例如，为一个具有 3 个类别的温度变量进行哑变量编码（比如，hot、medium 或者 cold），可以用 $(3-1)=2$ ，两个特征来进行设置，如下式所示：

$$\begin{aligned} \text{hot} &= \begin{cases} 1 & \text{如果 } x = \text{hot} \\ 0 & \text{其他} \end{cases} \\ \text{medium} &= \begin{cases} 1 & \text{如果 } x = \text{medium} \\ 0 & \text{其他} \end{cases} \end{aligned}$$

这里，只要知道 hot 和 medium 的值同时为 0 就足以说明温度是 cold，因此我们不需要为 cold 属性设置第 3 个特征。

哑变量编码的一个方便之处就在于哑变量编码的特征之间的距离总是为 1 或者 0，因此，与 min-max 标准化的数值型数据一样，这些值落在了一个相同的标度内，不需要进行额外的变换。



如果名义特征是有序的（可以将我们刚刚看到的温度变量作为例子），那么一种哑变量编码的替代方法就是给类别编号并且应用 min-max 标准化。例如，cold、warm 和 hot 可以编号为 1、2 和 3，min-max 标准化后为 0、0.5 和 1。使用该方法要注意的是，只有当你确信类别之间的步长相等时，才能应用该方法。例如，你可以证明，尽管 poor、middle class 和 wealthy 是有序的，但是 poor 和 middle class 之间的差异比 middle class 和 wealthy 之间的差异大（或小）。在这种情况下，哑变量编码是一种更保险的方法。

3.1.2 为什么 kNN 算法是懒惰的

基于近邻方法的分类算法被认为是懒惰学习算法，因为从技术上来说，没有抽象化的步骤。抽象过程和一般化过程都被跳过了，这就破坏了第 1 章给出的学习的定义。

从学习这个概念的严格定义上来说，懒惰学习并不是真正在学习什么。相反，它仅仅是一字不差地存储训练数据，这样训练阶段就进行得很快，同时伴随着一个潜在的不利因素，即进行预测的过程往往会变得相对较慢。由于高度依赖于训练案例（或实例），所以懒惰学习又称为**基于实例的学习**（instance-based learning）或者**机械学习**（rote learning）。

由于基于实例的学习算法并不会建立一个模型，所以该方法被归类为**非参数**（non-parametric）学习方法，即没有需要学习的数据参数。因为没有产生关于数据的理论，所以非参数方法限制了我们理解分类器如何使用数据的能力。另一方面，它允许学习算法发现数据中的自然模式，而不是试图将数据拟合为一个预先设定的形式。

尽管 kNN 分类器可能被认为是懒惰的，但它们还是很强大的，正如你不久就会看到的，kNN 的简单原则可以用于癌症的自动化筛查过程。

3.2 用 kNN 算法诊断乳腺癌

定期的乳腺癌检查使得疾病在引起明显的症状之前就得到诊断与治疗。早期的检测过程包括检查乳腺组织的异常肿块。如果发现一个肿块，那么就需要进行细针抽吸活检，即利用一根空心针从肿块中提取细胞的一小部分，然后临床医生在显微镜下检查细胞，从而确定肿块可能是恶性的还是良性的。

如果机器学习能够自动识别癌细胞，那么它将为医疗系统提供相当大的益处。自动化的过程很有可能提高检测过程的效率，从而可以让医生在诊断上花更少的时间，而在治疗疾病上花更多的时间。自动化的筛查系统还可能通过去除该过程中的内在主观人为因素来提供更高的检测准确性。

从带有异常乳腺肿块的女性身上的活检细胞的测度数据入手，应用 kNN 算法，从而研究机器学习用于检测癌症的功效。

3.2.1 第 1 步——收集数据

我们将使用来自 UCI 机器学习数据仓库（UCI Machine Learning Repository）的“威斯康星乳腺癌诊断”（Breast Cancer Wisconsin Diagnostic）数据集，该数据可以从网站 <http://archive.ics.uci.edu/ml> 获得。该数据是由威斯康星大学的研究者捐赠的，包括乳房肿块细针抽吸活检图像的数字化多项测度值，这些值代表出现在数字化图像中的细胞核的特征。

乳腺癌数据包括 569 例细胞活检案例，每个案例有 32 个特征。一个特征是识别号码，一个特征是癌症诊断结果，其他 30 个特征是数值型的实验室测量结果。癌症诊断结果用编码“M”表示恶性，用编码“B”表示良性。



想要阅读更多关于威斯康星乳腺癌数据的信息，可参考作者发表的论文：Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pp 861-870 by W.N. Street, W.H. Wolberg, and O.L. Mangasarian, 1993。

30 个数值型测量结果由数字化细胞核的 10 个不同特征的均值、标准差和最差值（即最大值）构成。这些特征包括：

- ☐ Radius（半径）
- ☐ Texture（质地）
- ☐ Perimeter（周长）
- ☐ Area（面积）
- ☐ Smoothness（光滑度）
- ☐ Compactness（致密性）
- ☐ Concavity（凹度）
- ☐ Concave points（凹点）
- ☐ Symmetry（对称性）
- ☐ Fractal dimension（分形维数）

根据它们的名字，所有的特征似乎都与细胞核的形状和大小有关。除非你是一个癌症医师，否则你不大可能知道每个特征如何与良性或者恶性肿瘤联系在一起。在我们继续机器学习的过程中，这些模式将会被揭示。

3.2.2 第 2 步——探索和准备数据

让我们来探索数据并且看看是否能让数据之间的关系明朗化一些。与此同时，我们要准备使用 kNN 学习算法所要用到的数据。



如果你计划跟着一起学习，那么你需要从 Packt 网站下载 `wisc_bc_data.csv` 文件，并将它保存到你的 R 工作目录下。本书中对该数据集做了非常轻微的修改。具体讲，增加了一个标题行，对行数据进行了随机排序。

与我们先前所做的一样，我们将从导入 CSV 数据文件开始，把威斯康星乳腺癌数据保存到数据框 `wbcd` 中：

```
> wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
```

正如我们所预期的那样，使用命令 `str(wbcd)` 可以确认数据是由 569 个案例和 32 个特征构成的。前几行的输出结果如下所示：

```
'data.frame': 569 obs. of 32 variables:
 $ id                : int  87139402 8910251 905520 ...
 $ diagnosis         : chr  "B" "B" "B" "B" ...
 $ radius_mean       : num  12.3 10.6 11 11.3 15.2 ...
```



```
$ texture_mean      : num  12.4 18.9 16.8 13.4 13.2 ...
$ perimeter_mean    : num  78.8 69.3 70.9 73 97.7 ...
$ area_mean         : num  464 346 373 385 712 ...
```

第一个变量是一个名为 `id` 的整型变量。由于这仅仅是每个病人在数据中唯一的标识符 (ID)，它并不能提供有用的信息，所以我们需要把它从模型中排除。



不管是什么机器学习方法，ID 变量总是要被剔除的，不这样做会导致错误的结果，因为 ID 可以用来独一无二地“预测”每一个案例。因此，包括标识符的模型很可能会受到过度拟合的影响，并且不太可能很好地推广到其他数据。

首先将 `id` 特征完全剔除。由于它位于第一列，所以我们可以复制一个不包括列 1 的 `wbcd` 数据框来剔除它：

```
> wbcd <- wbcd[-1]
```

接下来的变量是 `diagnosis`，它是我们特别感兴趣的，因为它是我们希望预测的结果。这个特征表示案例是来自于良性肿块还是恶性肿块。函数 `table()` 的输出结果表示 357 个肿块是良性的，而 212 个肿块是恶性的：

```
> table(wbcd$diagnosis)
  B   M
357 212
```

许多 R 机器学习分类器要求将目标属性编码为因子类型，所以我们需要重新编码 `diagnosis` 变量。同时，我们也会用 `labels` 参数对 B 值和 M 值给出含有更多信息的标签：

```
> wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),
                           labels = c("Benign", "Malignant"))
```

现在，当我们观察函数 `prop.table()` 的输出结果时，我们注意到，输出值被标记为 `Benign` 和 `Malignant`，分别有 62.7% 的良性肿块和 37.3% 的恶性肿块：

```
> round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)
      Benign Malignant
      62.7      37.3
```

其余的 30 个特征都是数值型的，与预期的一样，由 10 个细胞核特征的 3 种不同测量构成。作为示例，我们这里详细地观察 3 个特征：

```
> summary(wbcd[c("radius_mean", "area_mean", "smoothness_mean")])
      radius_mean      area_mean      smoothness_mean
Min.      : 6.981   Min.      :143.5   Min.      :0.05263
1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
Median :13.370   Median : 551.1   Median :0.09587
Mean    :14.127   Mean    : 654.9   Mean    :0.09636
3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
Max.    :28.110   Max.    :2501.0   Max.    :0.16340
```

纵观这些并排的特征，你注意到关于数值的一些问题吗？回想一下，kNN 的距离计算在

很大程度上依赖于输入特征的测量尺度。由于 `smoothness_mean` 的范围是从 0.05 ~ 0.16，而 `area_mean` 的范围是从 143.5 ~ 2501.0，所以在距离计算中，区域（`area-mean`）的影响比平滑度（`smoothness-mean`）的影响大很多，这可能会潜在地导致我们的分类器出现问题，所以我们应用 `min-max` 标准化方法重新调整特征的值得到一个标准的范围内。

1. 转换——`min-max` 标准化数值型数据

为了将这些特征 `min-max` 标准化，我们需要在 R 中创建一个 `normalize()` 函数，该函数输入一个数值型向量 `x`，并且对于 `x` 中的每一个值，减去 `x` 中的最小值再除以 `x` 中值的范围，最后，返回结果向量。该函数的代码如下：

```
> normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

运行上面的代码之后，函数 `normalize()` 就可以使用了。让我们用几个向量来测试这个函数：

```
> normalize(c(1, 2, 3, 4, 5))
[1] 0.00 0.25 0.50 0.75 1.00
> normalize(c(10, 20, 30, 40, 50))
[1] 0.00 0.25 0.50 0.75 1.00
```

该函数看来是能正确运行。事实上，尽管第二个向量中的值是第一个向量中的值的 10 倍，但是在 `min-max` 标准化以后，这两个向量返回的结果是完全一样的。

现在，我们可以将 `normalize()` 函数应用于我们数据框中的数值型特征。我们并不需要对这 30 个数值型变量逐个进行 `min-max` 标准化，这里可以使用 R 中的一个函数来自动完成此过程。

R 中的 `lapply()` 函数可以输入一个列表，然后把一个函数应用到列表的每一个元素。因为数据框是一个含有等长度向量的列表，所以我们可以使用 `lapply()` 函数将 `normalize()` 函数应用到数据框中的每一个特征。最后一个步骤是，应用函数 `as.data.frame()` 把 `lapply()` 返回的列表转换成一个数据框。全部过程如下所示：

```
> wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
```

以通俗的语言来讲，该命令把 `normalize()` 函数应用到数据框 `wbcd` 的第 2 ~ 31 列，把产生的结果列表转换成一个数据框，并给该数据框赋予一个名称 `wbcd_n`。这里使用的后缀 “_n” 是作为一个提示，即 `wbcd` 中的值已经被 `min-max` 标准化了。

为了确认转换是否正确应用，让我们来看看其中一个变量的汇总统计量：

```
> summary(wbcd_n$area_mean)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0000  0.1174   0.1729   0.2169   0.2711   1.0000
```

正如预期的那样，`area_mean` 变量的原始范围是 143.5 ~ 2501.0，而现在的范围是 0 ~ 1。

2. 数据准备——创建训练数据集和测试数据集

尽管所有的 569 个活检的良性或者恶性情形都已被标记，但是预测我们已经知道的结果并不是特别令人感兴趣的。此外，我们在训练期间得到的算法分类好坏的衡量指标可能是有误的，因为我们不知道数据发生过度拟合的程度，或者说，不知道推广到未知情形时的效果有多好。一个更有趣的问题是，对于一个没有标记数据的数据集，学习算法的性能怎样。如果我们有机会使用实验室，那么我们可以将学习算法应用到接下来的 100 个未知癌症情形的肿块的测量数据，并且看看与用传统方法得到的诊断结果相比，机器学习算法的预测怎样。

由于缺少这样的数据，所以我们可以通过把我们的数据划分成两部分来模拟这种方案：一部分是用来建立 kNN 模型的训练数据集；另一部分是用来估计模型预测准确性的测试数据集。我们使用前 469 条记录作为训练数据集，剩下的 100 条记录用来模拟新的病人。

使用在第 2 章中给出的数据提取方法，我们将把 wdbc_n 数据框拆分为 wbcd_train 数据框和 wbcd_test 数据框：

```
> wbcd_train <- wbcd_n[1:469, ]
> wbcd_test <- wbcd_n[470:569, ]
```

如果上面的代码让你困惑了，那么记住，从数据框中提取数据使用的是 [row, column] 语法，如果行值或者列值是空的，就表明所有的行或者列都应该被包含在内。因此，第一行代码取的是第 1 ~ 469 行的所有列，第二行取的是 470 ~ 569 行的 100 行的所有列。



当构造训练数据集和测试数据集时，保证每一个数据集都是数据全集的一个有代表性的子集是很重要的。在刚刚看到的案例中，记录已经按照随机顺序排列，所以我们可以简单地提取 100 个连续的记录来创建一个测试数据集。如果数据是按照非随机的模式排列的，比如按时间顺序或者以具有相似值的组的顺序，那么这将不是合适的创建上述两个数据集的方法。在这些情况下，需要用到随机抽样方程。

当我们构建训练数据和测试数据时，我们剔除了目标变量 diagnosis。为了训练 kNN 模型，我们需要把这些类的标签存储在一个因子类型的向量中，然后把该向量划分为训练数据集和测试数据集：

```
> wbcd_train_labels <- wbcd[1:469, 1]
> wbcd_test_labels <- wbcd[470:569, 1]
```

该代码使用 wbcd 数据框第一列的 diagnosis 因子，并且创建了 wbcd_train_labels 和 wbcd_test_labels 两个向量。我们将会下面的分类器的训练和评估步骤中使用这些向量。

3.2.3 第 3 步——基于数据训练模型

有了训练数据集和标签向量后，我们现在准备好对未知记录进行分类。对于 kNN 算法，训练阶段实际上不包括模型的建立——训练一个懒惰学习算法的过程，就像 kNN 算法仅涉及以结构化格式存储输入的数据。

为了将我们的测试实例进行分类，我们使用来自 `class` 添加包的一个 `kNN` 算法实现，该添加包提供了一组用于分类的基本 R 函数。如果该添加包尚未安装到你的系统上，你可以通过输入以下命令来安装它：

```
> install.packages("class")
```

在你希望使用这些函数的任何会话期间，你只需要输入命令 `library(class)`，就可加载该添加包。

`class` 添加包中的 `knn()` 函数提供了一个标准的 `kNN` 算法实现。对于测试数据中的每一个实例，该函数将使用欧氏距离标识 k 个近邻，其中 k 是用户指定的一个数。于是，通过 k 个近邻的“投票”来对测试个案进行分类——确切地说，该过程涉及将实例归类到 k 个近邻中的大多数所在的那个类。如果各个类的票数相等，该测试实例会被随机分类。



在其他 R 添加包中，还有几个其他的 `kNN` 函数，提供了更加复杂或者更加高效的算法实现。如果你受到 `knn()` 函数的限制，可以查看[综合R档案网络](#) (Comprehensive R Archive Network, CRAN)，看看是否有更多信息。虽然如此，你还可能对这里的基本函数 `knn()` 的良好工作而感到惊讶。

使用 `knn()` 函数进行训练和分类是在一个单一的函数调用中执行的，它包含 4 个参数，如下表所示。

kNN 分类语法
应用 <code>class</code> 添加包中的函数 <code>knn()</code>
<p>创建分类器并进行预测：</p> <pre>p <- knn(train, test, class, k)</pre> <ul style="list-style-type: none"> • <code>train</code>: 一个包含数值型训练数据的数据框 • <code>test</code>: 一个包含数值型测试数据的数据框 • <code>class</code>: 包含训练数据每一行分类的一个因子向量 • <code>k</code>: 标识最近邻数目的一个整数 <p>该函数返回一个因子向量，该向量含有测试数据框中每一行的预测分类。</p> <p>例子：</p> <pre>wbcd_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels, k = 3)</pre>

我们已经有了把 `kNN` 算法应用到该数据集中的几乎所有参数。我们把数据划分成训练数据集和测试数据集，每个数据集都有完全相同的数值特征。训练数据中的标签存储在一个单独的因子向量中，唯一剩下的参数是 k ，它指定投票中所包含的邻居数量。

由于训练数据集含有 469 个实例，所以我们可能会尝试 $k = 21$ ，它是一个大约等于 469 的平方根的奇数。使用奇数将会减少各个类票数相等这一情况发生的可能性。

现在，我们可以使用 `knn()` 函数来给测试数据进行分类：

```
> wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                        cl = wbcd_train_labels, k=21)
```

函数 `knn()` 返回一个因子向量，为测试数据集中的每一个案例返回一个预测标签，我们将该因子向量命名为 `wbcd_test_pred`。

3.2.4 第4步——评估模型的性能

该过程的下一步就是评估 `wbcd_test_pred` 向量中预测的分类与 `wbcd_test_labels` 向量中已知的值的匹配程度如何。为了做到这一点，我们可以使用 `gmodels` 添加包中的 `CrossTable()` 函数，它在第2章中介绍过。如果你还没有安装该添加包，可以使用命令 `install.packages("gmodels")` 进行安装。

在使用 `library(gmodels)` 命令加载该添加包后，可以创建一个用来标识两个向量之间一致性的交叉表。指定参数 `prop.chisq = FALSE`，将会从输出中去除不需要的卡方 (chi-square) 值，如下所示：

```
> CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
             prop.chisq=FALSE)
```

由此产生的表如下所示。

Cell Contents

			N
	N / ROW Total		
	N / Col Total		
	N / Table Total		

Total observations in Table: 100

wbcd_test_labels	wbcd_test_pred		
	Benign	Malignant	Row Total
Benign	61 1.000 0.968 0.610 TN	0 0.000 0.000 0.000 FP	61 0.610
Malignant	2 0.051 0.032 0.020 FN	37 0.949 1.000 0.370 TP	39 0.390
Column Total	63 0.630	37 0.370	100

表格中单元格的百分比表示落在4个分类里的值所占的比例。在左上角的单元格（标记为 **TN**）中，是真阴性（True Negative）的结果。100个值中有61个值标识肿块是良性的，而 `kNN` 算法也正确地把它标识为良性的。在右下角的单元格（标记为 **TP**）中，显示的是真阳性（True Positive）的结果，这里表示的是分类器和临床确定的标签一致认为肿块是恶性的情形。100个预测值中有37个是真阳性（True Positive）的。

落在另一条对角线上的单元格包含了 `kNN` 算法与真实标签不一致的案例计数。位于左下角 **FN** 单元格的2个案例是假阴性（False Negative）的结果。在这种情况下，预测的值是良性的，但肿瘤实际上是恶性的。这个方向上的错误可能会产生极其高昂的代价，因为它们可能导致一位病人认为自己没有癌症，而实际上这种疾病可能会继续蔓延。如果右上角标记

为 FP 的单元格里有值，它包含的是假阳性（False Positive）的结果。当模型把肿块标识为恶性的，而事实上它是良性时就会产生这里的值。尽管这类错误没有假阴性（False Negative）的结果那么危险，但这类错误也应该避免，因为它们可能会导致医疗系统的额外财政负担，或者病人的额外压力，毕竟这需要提供额外的检查或者治疗。



如果我们需要，我们可以通过将每一个肿块分类为恶性肿块来完全排除假阴性（False Negative）的结果。显然，这是一个不切实际的策略。然而，它说明了一个事实，即预测涉及假阳性（False Positive）比率和假阴性（False Negative）比率之间的一个平衡。在第 10 章中，你将学习更复杂的方法来度量预测的准确性，根据每种错误类型的成本，这些方法可以用来找出那些错误率可以被优化的地方。

一共有 2%，即根据 kNN 算法，100 个肿块中，有 2 个是被错误分类的。虽然对于仅用几行的 R 代码，就得到 98% 的准确度似乎令人印象深刻，但是我们可以尝试一些其他的模型迭代方法来看看我们是否可以提高性能并减少错误分类值的数量，特别当错误是危险的假阴性（False Negative）结果时。

3.2.5 第 5 步——提高模型的性能

对于前面的分类器，我们将尝试两种简单的改变。第一，我们将使用另一种方法重新调整数值特征；第二，我们将尝试几个不同的 k 值。

1. 转换——z-score 标准化

虽然 min-max 标准化是传统上用于 kNN 分类的方式，但它并不一定总是最合适的调整特征的方法。因为 z-score 标准化后的值没有预定义的最小值和最大值，所以极端值不会被压缩到中心。有人可能会怀疑在有一个恶性肿瘤的情况下，可能会得到一些非常极端的异常值，因为肿瘤的生长不受控制。然而，让异常值在距离计算中占有更大的权重有可能是合理的。让我们来看看 z-score 标准化是否能够提高预测的准确性。

为了标准化一个向量，我们可以使用 R 内置的 `scale()` 函数，该函数默认使用 z-score 标准化来重新调整特征的值。`scale()` 函数提供的一个额外好处就是它能够直接应用于数据框，这样，我们可以避免使用 `lapply()` 函数。为了创建一个 wbcd 数据的 z-score 标准化版本，我们可以使用下面的命令，该命令重新调整除了 `diagnosis` 以外的所有特征，并且以数据框的形式把结果存储在 `wbcd_z` 变量中，后缀 `_z` 作为一个提示，即特征的值已经进行了 z-score 标准化。

```
> wbcd_z <- as.data.frame(scale(wbcd[-1]))
```

为了确认转换是否正确，我们可以看一看汇总统计量：

```
> summary(wbcd_z$area_mean)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
-1.4530 -0.6666 -0.2949 0.0000 0.3632 5.2460
```

一个 z-score 标准化后的变量的均值应该始终为 0，而且其值域应该非常紧凑，一个大于 3 或者小于 -3 的 z-score 表示一个极其罕见的值，上面的汇总似乎是合理的。

正如我们之前所做的那样，我们需要将数据划分为训练数据集和测试数据集，然后使用 knn() 函数对测试实例进行分类，最后，我们使用 CrossTable() 函数来比较预测的标签和实际的标签：

```
> wbcd_train <- wbcd_z[1:469, ]
> wbcd_test <- wbcd_z[470:569, ]
> wbcd_train_labels <- wbcd[1:469, 1]
> wbcd_test_labels <- wbcd[470:569, 1]
> wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test,
                        cl = wbcd_train_labels, k=21)
> CrossTable(x = wbcd_test_labels, y = wbcd_test_pred,
             prop.chisq=FALSE)
```

不幸的是，在下面的表格中，应用新的变换得到的结果的准确性略有降低。之前，我们正确分类了 98% 的案例，而这一次我们仅正确分类了 95% 的案例。更糟糕的是，我们并没有在假阴性 (False Negative) 的分类结果上做得更好。

wbcd_test_labels	wbcd_test_pred		Row Total
	Benign	Malignant	
Benign	61	0	61
	1.000	0.000	
	0.924	0.000	
	0.610	0.000	
Malignant	5	34	39
	0.128	0.872	
	0.076	1.000	
	0.050	0.340	
Column Total	66	34	100
	0.660	0.340	

2. 测试其他的 k 值

或许，我们可以做得更好，即检验应用不同的 k 值模型的性能。使用 min-max 标准化的训练数据集和测试数据集，然后用几个不同的 k 值来对相同的 100 个记录进行分类。每次迭代的假阴性 (False Negative) 和假阳性 (False Positive) 的数量如下表所示

k 值	假阴性的数量	假阳性的数量	错误分类的比例
1	1	3	4%
5	2	0	2%
11	3	0	3%
15	3	0	3%
21	2	0	2%
27	4	0	4%

虽然分类器永远不会很完美，但是 1NN 算法能够避免一些假阴性（False Negative）的结果，不过是以增加假阳性（False Positive）的结果为代价。然而，重要的是记住，为了过于准确预测测试数据来调整我们的方法是不明智的，毕竟，一组不同的 100 位病人记录很可能与那些用来测量我们模型性能的记录有所不同。



如果你需要确认一个学习算法能推广到未来的数据，那么你可能需要随机地创建几组 100 位病人的记录，并且在这些数据上重复测试模型结果。第 10 章将深入讨论仔细评估机器学习模型性能的方法。

3.3 总结

在本章中，我们学习了使用 k 近邻进行分类。不同于很多其他的分类算法， kNN 并没有进行任何学习，它只需要逐字存储训练数据。然后使用一个距离函数将未标记的测试案例与训练数据集中最相似的记录进行匹配，并将未标记案例的邻居的标签分配给它。

尽管事实上 kNN 是一个简单的算法，但是它却能够处理极其复杂的任务，比如识别癌细胞的肿块。用简单的几行 R 代码，就能够以高达 98% 的正确率来识别一个肿块是恶性的还是良性的。

在第 4 章中，我们将研究使用概率来估计一个观测值落入某些特定类别中的分类方法，比较该方法与 kNN 算法有何不同将会很有趣。之后，在第 9 章中，我们将学习一个与 kNN 算法很相似的算法，该方法把距离度量用于一个完全不同的学习任务中。



概率学习——朴素贝叶斯分类

当一位气象学家提供天气预报时，通常会使用像“70%的可能性会下雨”这样的术语来预测降雨，这些预测称为下雨的概率。你有没有想过他们是如何计算的呢？这是一个让人困惑的问题，因为在现实生活中，要么下雨，要么不下雨。

这些估计都是基于概率方法或者关于描述不确定性的方法得到的。他们通过对过去已发生事件的数据信息来推断未来的事件。在天气预报这个例子中，下雨的可能性描述了通过类似测量大气条件的方法得到的前几天下雨的概率。因此，70%下雨的可能性意味着，在过去有类似天气特征的 10 个例子中，有 7 次在该地区的某个地方下雨了。

本章讲述了一种机器学习算法，即依据概率原则进行分类的**朴素贝叶斯算法**。正如气象学家预测天气一样，朴素贝叶斯算法就是应用先前事件的有关数据来估计未来事件发生的概率。举个例子，朴素贝叶斯的一个常见应用就是根据过去垃圾邮件中单词使用的频率来识别新的垃圾邮件。在学习朴素贝叶斯是如何应用时，将学习：

- 用于朴素贝叶斯的基本概率原则。
- 基于 R，用专门的方法、可视化和数据结构分析文本数据。
- 如何运用朴素贝叶斯分类器建立短信过滤器并通过 R 实现。

如果你之前已经上过统计学课程，本章中的一些材料看上去可能是对统计学学科的一些回顾。即便如此，这也将有助于你重新整理概率知识，而且这些原则也是“朴素贝叶斯”如何得到这样一个奇怪名称的依据。

4.1 理解朴素贝叶斯

已经存在了几百年的基本统计学思想对理解朴素贝叶斯算法是必要的，该算法起源于 18

世纪数学家托马斯·贝叶斯 (Thomas Bayes) 的工作, 托马斯·贝叶斯发明了用于描述事件的概率以及如何根据附加信息修正概率的基本数学原理 (现在称为**贝叶斯方法**)

以后, 我们将会进一步深入研究, 但现在知道如下事实就够了: 一个事件发生的概率是一个介于 0 ~ 100% 之间的数。在给定现有证据的条件下, 这个数给出了一个事件将发生的可能性。概率越小, 事件发生的可能性就越小。概率为 0 表示事件绝对不会发生, 而概率为 100% 表示事件肯定会发生。

基于贝叶斯方法的分类器是利用训练数据并根据特征的取值来计算每个类别被观察到的概率。当分类器之后被应用到无标签数据时, 分类器就会根据观测到的概率来预测新的特征最有可能属于哪个类。这是简单的想法, 但根据这种想法就产生了一种方法, 这种方法得到的结果与很多复杂算法得到的结果是等价的。事实上, 贝叶斯分类器已用于以下方面:

- 文本分类, 比如垃圾邮件过滤、作者识别和主题分类等
- 在计算机网络中进行入侵检测或者异常检测
- 根据一组观察到的症状, 诊断身体状况

通常情况下, 贝叶斯分类器最适用于解决这样一类问题: 在这类问题中, 为了估计一个结果的概率, 从众多属性中提取的信息应该被同时考虑。尽管很多算法忽略了具有弱影响的一些特征, 但是贝叶斯方法利用了所有可以获得的证据来巧妙地修正预测。如果有大量特征产生的影响较小, 但将它们放在一起, 它们的组合影响可能会相当大。

4.1.1 贝叶斯方法的基本概念

在进入朴素贝叶斯算法学习之前, 我们值得花一些时间来定义一些概念, 这些概念在贝叶斯方法中经常用到。用一句话概括, 贝叶斯概率理论植根于这样一个思想, 即一个事件的似然估计应建立在手中已有证据的基础上。**事件 (event)** 就是可能的结果, 比如晴天和阴雨天、抛掷一枚硬币得到的正面或者反面、垃圾电子邮件和非垃圾电子邮件等。**试验 (trial)** 就是事件发生一次的机会, 比如某天的天气、抛掷一枚硬币、一封电子邮件等。

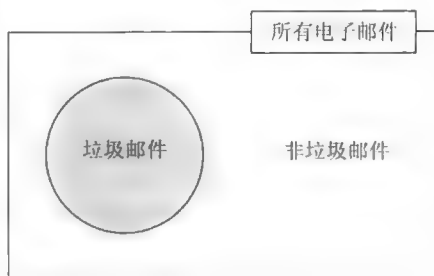
1. 概率

一个事件发生的**概率**可以通过观测到的数据来估计, 即用该事件发生的试验的次数除以试验的总次数。例如, 如果 10 天中有 3 天下雨了, 那么就可以估计下雨的概率为 30%。同样, 如果 50 封电子邮件中有 10 封是垃圾邮件, 那么可以估计垃圾邮件的概率为 20%。通常用符号 $P(A)$ 来表示事件 A 发生的概率, 比如 $P(\text{垃圾邮件})=0.20$ 。

一个试验的所有可能结果的概率之和一定为 100%。因此, 如果试验只有两个不可能同时发生的结果, 比如硬币的正面和反面、垃圾邮件和非垃圾邮件, 知道了其中一个结果发生的概率就意味着知道了另一个结果发生的概率。例如, 给定 $P(\text{垃圾邮件})=0.20$, 我们就能够计算出 $P(\text{非垃圾邮件})=1-P(\text{垃圾邮件})=1-0.20=0.80$ 。之所以可以这样计算, 是因为垃圾邮件和非垃圾邮件是两个完全相互独立的事件, 这意味着垃圾邮件和非垃圾邮件两个事件不可能在同一时间发生, 它们是试验仅有的两个可能的结果。为了简记, 用符号 $P(\neg A)$ 来表示

事件 A 不发生的概率, 例如 $P(\text{一垃圾邮件})=0.80$ 。

作为示例, 将事件概率想象为一个两维空间是有益的, 该空间被分割为不同事件的概率。在下面的图中, 矩形代表电子邮件所有可能发生的结果的集合, 其中的圆代表垃圾电子邮件的概率, 其余的 80% 代表不是垃圾电子邮件的概率。

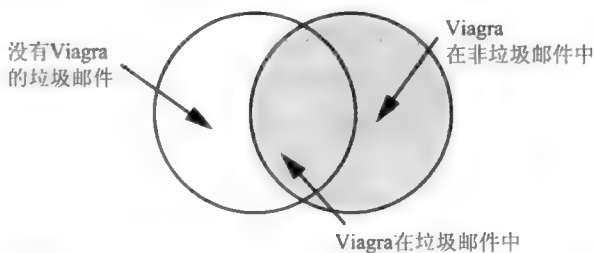
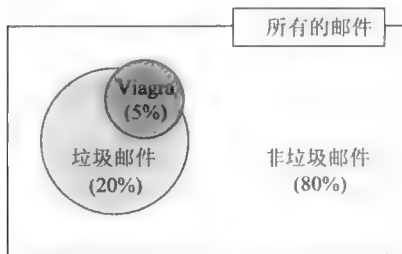


2. 联合概率

通常, 对于同一试验, 我们感兴趣的是对几个非互斥事件的研究。如果一些发生的事件中有我们感兴趣的事件, 或许我们可以用它们来进行预测。思考这样一个例子, 第二个事件的发生建立在电子邮件包含单词 Viagra 事件发生的条件下。对于大多数人来说, 这个单词只可能出现在垃圾邮件中, 因此, 在信息中出现这个单词是说明该电子邮件是垃圾邮件的一个非常有力的证据。鉴于本次事件的发生, 更新上一幅图形, 可能得到如下图左边的图形。

注意, 在该图中, 含有单词 Viagra 的圆没有全部落在垃圾邮件的圆中, 也没有完全包含垃圾邮件的圆。这表明, 不是所有的垃圾邮件都含有单词 Viagra, 也不是含有单词 Viagra 的邮件就一定是垃圾邮件。

为了仔细观察垃圾邮件和含有单词 Viagra 的邮件之间的重叠, 我们将其放大, 应用可视化的文氏图 (Ven Diagram) 该图在 19 世纪后期由约翰·维恩 (John Venn) 首先使用, 该图用圆来说明事件组之间的重叠。在大多数文氏图中, 比如下右图, 圆的大小和重叠的程度并不重要, 只是用这样一种方法来提醒我们对所有可能的事件组合来分配概率。



我们知道垃圾邮件占有所有电子邮件的 20% (左边的圆), 含有单词 Viagra 的邮件占有所有电子邮件的 5% (右边的圆)。我们的工作就是量化这两个比例之间的重叠程度, 换句话说, 我们希望估计 $P(\text{垃圾邮件})$ 和 $P(\text{Viagra})$ 同时发生的概率, 记为 $P(\text{垃圾邮件} \cap \text{Viagra})$ 。

概率 $P(\text{垃圾邮件} \cap \text{Viagra})$ 的计算取决于这两个事件的联合概率, 即如何将一个事件发生的概率和另一个事件发生的概率联系在一起。如果这两个事件完全不相关, 我们称为独立事件。例如, 抛硬币的结果与天气是晴天还是阴雨天是相互独立的。

如果所有的事件都是相互独立的, 利用已经获得的另一个事件的数据将不可能预测任何一个事件发生的概率, 另一方面, 相关事件是建立预测模型的基础。例如, 根据云的存在,

很有可能预测是一个雨天；根据单词 Viagra 的出现，预测电子邮件是一封垃圾邮件。

如果我们知道 $P(\text{垃圾邮件})$ 和 $P(\text{Viagra})$ 是相互独立的，则很容易计算 $P(\text{垃圾邮件} \cap \text{Viagra})$ ，即这两个事件同时发生的概率。由于在所有电子邮件中，20% 为垃圾邮件，5% 的邮件含有单词 Viagra，所以我们可以计算 20% 中的 5% ($0.05 \times 0.20 = 0.01$)，即含有单词 Viagra 的垃圾邮件占有所有电子邮件的 1%。更一般地，对于独立事件 A 和事件 B ，这两个事件同时发生的概率 $P(A \cap B) = P(A) \times P(B)$ 。

在现实中， $P(\text{垃圾邮件})$ 和 $P(\text{Viagra})$ 更可能是高度相关的，因此上述计算是不正确的，我们需要一个精确的公式来描述这两个事件之间的关系。

3. 基于贝叶斯定理的条件概率

相关事件之间的关系可以用贝叶斯定理来描述，如下面的公式所示。符号 $P(A|B)$ 表示在事件 B 已经发生的条件下，事件 A 发生的概率。这就是条件概率，因为事件 A 发生的概率依赖于事件 B 的发生（即条件）。

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(A \cap B)}{P(B)}$$

为了解贝叶斯定理在实际中的应用，可以假设你的任务是估算收到的电子邮件是垃圾邮件的概率。在没有任何附加证据的条件下，最合理的猜测就是事先收到垃圾邮件的概率，即前面例子中的 20%。这个估计称为先验概率。

现在，假设你获得了一条额外的证据，你被告知收到的电子邮件使用了单词 Viagra，在先前的垃圾邮件中出现单词 Viagra 的概率称为似然概率（likelihood），而单词 Viagra 出现在任何一封邮件中的概率称为边际似然概率（marginal likelihood）。

将贝叶斯定理应用到这条额外的证据上，我们可以计算后验概率（posterior），这个概率用来衡量该邮件是垃圾邮件的可能性。如果计算出的后验概率远大于 50%，则该信息更可能是垃圾信息，应该过滤掉。下面的公式就是对于给定证据的贝叶斯定理：

$$P(\text{spam} | \text{Viagra}) = \frac{P(\text{Viagra} | \text{spam})P(\text{spam})}{P(\text{Viagra})}$$

似然概率 ↗
先验概率 ↘

后验概率 ↗
↘ 边际似然概率

为了计算贝叶斯定理中每一个组成部分的概率，我们必须构造一个频率表（如下面的左图所示），该表记录了单词 Viagra 出现在垃圾邮件和非垃圾邮件中的次数。与双向交叉列表相似，表的一个维度表示分类变量的水平（垃圾邮件或者非垃圾邮件），而另一个维度表示特征的水平（即单词 Viagra 是否出现：Yes 或 No），表中的元素表示具有分类值和特征值特定组合的实例的数目。根据频率表，可以构造似然表，如下面的右图所示。

Viagra			
频数	Yes	No	总计
垃圾邮件	4	16	20
非垃圾邮件	1	79	80
总计	5	95	100

Viagra			
似然	Yes	No	总计
垃圾邮件	4/20	16/20	20
非垃圾邮件	1/80	79/80	80
总计	5/100	95/100	100

根据似然表,可以得到 $P(\text{Viagra} | \text{垃圾邮件}) = 4/20 = 0.20$, 这意味着在垃圾邮件中, 含有单词 Viagra 的邮件的概率为 20%。此外, 根据定理 $P(B|A) \times P(A) = P(A \cap B)$, 我们可以计算 $P(\text{垃圾邮件} \cap \text{Viagra})$, 即

$$P(\text{Viagra} | \text{垃圾邮件}) \times P(\text{垃圾邮件}) = (4/20) \times (20/100) = 0.04$$

此次的估计值是先前在错误的独立性假设下估计值的 4 倍, 它说明了贝叶斯定理在计算联合概率中的重要性。

为了计算后验概率 $P(\text{垃圾邮件} | \text{Viagra})$, 我们利用贝叶斯定理, 即

$$P(\text{垃圾邮件} | \text{Viagra}) = P(\text{Viagra} | \text{垃圾邮件}) \times P(\text{垃圾邮件}) / P(\text{Viagra}) = (4/20) \times (20/100) / (5/100) = 0.8$$

因此, 如果电子邮件含有单词 Viagra, 那么该电子邮件是垃圾邮件的概率为 80%。所以, 任何含有单词 Viagra 的消息都需要被过滤掉。

这就是商业垃圾邮件过滤器的工作方式, 尽管在计算频率表和似然表时会同时考虑更多数目的词语。在下一节中, 我们将看到当有额外的特征时, 这一概念是如何被使用的。

4.1.2 朴素贝叶斯算法

朴素贝叶斯 (Naive Bayes, NB) 算法描述应用贝叶斯定理进行分类的一个简单应用。尽管这不是唯一应用贝叶斯方法的机器学习方法, 但它是最常见的, 尤其在文本分类应用中已经成为一种准则。该算法的优缺点如下表所示。

优点	缺点
<ul style="list-style-type: none"> 简单、快速、有效 	<ul style="list-style-type: none"> 依赖于一个常用的错误假设, 即一样的重要性和独立特征
<ul style="list-style-type: none"> 能处理好噪声数据和缺失的数据 	<ul style="list-style-type: none"> 应用在含有大量数值特征的数据集时并不理想
<ul style="list-style-type: none"> 需要用来训练的例子相对较少, 但同样能处理好大量的例子 	<ul style="list-style-type: none"> 概率的估计值相对于预测的类而言更加不可靠
<ul style="list-style-type: none"> 很容易获得一个预测的估计概率值 	

朴素贝叶斯算法之所以这样命名是因为关于数据有一对“简单”的假设。特别地, 朴素贝叶斯假设数据集的所有特征都具有相同的重要性和独立性, 而在大多数的实际应用中, 这些假设是鲜有成立的。

举个例子, 假设你试图通过监控电子邮件来识别垃圾邮件, 那么几乎可以肯定, 邮件中的某些特征比其他特征更重要。比如, 相对邮件内容来说, 电子邮件的发件人是判别垃圾邮件的一个更重要的指标。而且, 出现在邮件主体中的词和主体中的其他词并不是相互独立

的，因为有些词的出现正好暗示着其他词很可能出现。一封含有单词 **Viagra** 的邮件有极大的可能包含单词 **prescription** 或者 **drug**。

然而，在大多数情况下，当违背这些假设时，朴素贝叶斯依然可以很好地应用，甚至在极端事件中，特征之间具有很强的依赖性时，朴素贝叶斯也可以用。由于该算法的通用性和准确性，适用于很多类型的条件，所以在分类学习任务中，朴素贝叶斯算法往往是很强大的，排在候选算法的第一位。



为什么在错误的假设条件下，朴素贝叶斯算法还能有效应用呢？关于其准确性的原因存在很多推测，一种解释是：只要预测的分类值是准确的，那么获得精确的概率估计值并不重要。例如，如果垃圾邮件过滤器能正确识别垃圾邮件，那么在其预测时，它是有 51% 的把握，还是有 99% 的把握，这还重要吗？关于这一节更多的信息，可参阅 Pedro Domingos 和 Michael Pazzani 关于机器学习的文献“On the optimality of the simple Bayesian classifier under zero-one loss in Machine Learning” (1997)

1. 朴素贝叶斯分类

我们通过增加对词语 **Money**、**Groceries** 和 **Unsubscribe** 的监测来改善我们的垃圾邮件过滤器。我们可以通过构建出现的这 4 个单词（记为 W_1 、 W_2 、 W_3 和 W_4 ）的似然表来训练朴素贝叶斯算法，对 100 封电子邮件分析后的似然表如下表所示。

	Viagra(W_1)		Money(W_2)		Groceries(W_3)		Unsubscribe(W_4)		
似然	Yes	No	Yes	No	Yes	No	Yes	No	总计
垃圾邮件	4/20	16/20	10/20	10/20	0/20	20/20	12/20	8/20	20
非垃圾邮件	1/80	79/80	14/80	66/80	8/80	71/80	23/80	57/80	80
总计	5/100	95/100	24/100	76/100	8/100	91/100	35/100	65/100	100

在收到新的消息后，给定文本信息中这些单词的似然，必须通过计算后验概率来确定这些消息更像是垃圾邮件还是非垃圾邮件。例如，有一条消息包含单词 **Viagra** 和 **Unsubscribe**，但是不包含 **Money** 和 **Groceries**。

利用贝叶斯定理，我们可以定义这个问题的概率——一封邮件在给定 $Viagra = \text{Yes}$ 、 $Money = \text{No}$ 、 $Groceries = \text{No}$ 和 $Unsubscribe = \text{Yes}$ 条件下为垃圾邮件的概率，如下面公式所示。

$$P(\text{垃圾邮件} | W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) = \frac{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4 | \text{垃圾邮件}) P(\text{垃圾邮件})}{P(W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4)}$$

有很多原因使这个公式在计算上难以解决。由于额外特征信息的增加，需要巨大的内存来存储所有可能的交叉事件的概率，想象出现 4 个单词事件的文氏图的复杂性，更不用说数百个甚至更多事件发生的复杂性，为此，我们需要巨大的训练数据集确保有足够的数据来对所有的可能交叉事件建模。

如果我们可以利用朴素贝叶斯中事件独立性的假设,那么计算将会变得很简单,具体来说,朴素贝叶斯假设**类条件独立**(class-conditional independence),这意味着只要事件在同类取值的条件下,这些事件就是相互独立的。条件独立性的假设允许我们应用独立事件的概率原则来简化计算公式。此时,你可能会想到公式 $P(A \cap B) = P(A) \times P(B)$,于是我们可以得到一个更加容易计算的公式,如下所示:

$$\begin{aligned} & P(\text{垃圾邮件} | W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \\ = & \frac{P(W_1 | \text{垃圾邮件}) P(\neg W_2 | \text{垃圾邮件}) P(W_3 | \text{垃圾邮件}) P(W_4 | \text{垃圾邮件}) P(\text{垃圾邮件})}{P(W_1) P(\neg W_2) P(\neg W_3) P(W_4)} \end{aligned}$$

与此公式进行比较,我们可以得到关于非垃圾邮件概率的公式:

$$\begin{aligned} & P(\text{非垃圾邮件} | W_1 \cap \neg W_2 \cap \neg W_3 \cap W_4) \\ = & \frac{P(W_1 | \text{非垃圾邮件}) P(\neg W_2 | \text{非垃圾邮件}) P(W_3 | \text{非垃圾邮件}) P(\neg W_4 | \text{非垃圾邮件}) P(\text{非垃圾邮件})}{P(W_1) P(\neg W_2) P(\neg W_3) P(W_4)} \end{aligned}$$

利用似然表中的数据,我们就可以为这些公式填充数值,由于这两个公式中的分母是一样的,所以现在可以忽略它。垃圾邮件的总似然为:

$$(4/20) \times (10/20) \times (20/20) \times (12/20) \times (20/100) = 0.012$$

在给定模型下,非垃圾邮件的总似然为:

$$(1/80) \times (66/80) \times (71/80) \times (23/80) \times (80/100) = 0.002$$

因为 $0.012 / 0.002 = 6$,所以我们可以认为该消息是垃圾邮件的可能性是非垃圾邮件的6倍,即更有可能是垃圾邮件。然而,将这些数值转换成概率,我们还需要最后一步。

该消息是垃圾邮件的概率等于该消息是垃圾邮件的似然除以该消息是垃圾邮件或非垃圾邮件的总似然,即

$$0.012 / (0.012 + 0.002) = 0.857$$

同样,该消息是非垃圾邮件的概率等于该消息是非垃圾邮件的似然除以该消息是垃圾邮件或非垃圾邮件的总似然,即

$$0.002 / (0.012 + 0.002) = 0.143$$

给定该消息中4个单词出现的情况,我们期望该消息是垃圾邮件的概率为85.7%,是非垃圾邮件的概率为14.3%,因为这两个事件是完全斥事件,所以它们的概率之和为1。

我们在前面的例子中使用的朴素贝叶斯分类算法可以总结为如下的公式。从本质上讲,在给定特征 F_1 到 F_n 提供的证据的条件下,类 C 中水平为 L 的概率等于每一条证据在类 C 的水平 L 下的条件概率的乘积,再乘以类 C 的水平 L 的先验概率和尺度因子 $1/Z$,尺度因子 $1/Z$ 将把上述结果转换为一个概率值:

$$P(C_L | F_1, \dots, F_n) = \frac{1}{Z} P(C_L) \prod_{i=1}^n P(F_i | C_L)$$

2. 拉普拉斯估计

让我们再看一个例子，假设我们收到另一条消息，这一次该消息包含的单词有：Viagra、Money、Groceries 和 Unsubscribe。利用之前的贝叶斯算法，我们可以计算垃圾邮件的似然如下：

$$(4/20) \times (10/20) \times (0/20) \times (12/20) \times (20/100) = 0$$

非垃圾邮件的似然为：

$$(1/80) \times (14/80) \times (8/80) \times (23/80) \times (80/100) = 0.000\ 05$$

因此，该消息是垃圾邮件的概率为：

$$0 / (0 + 0.000\ 05) = 0$$

该消息是非垃圾邮件的概率为：

$$0.000\ 05 / (0 + 0.000\ 05) = 1$$

这些结果表明该消息是垃圾邮件的概率为 0，是非垃圾邮件的概率为 100%。这样的预测结果有意义吗？很可能没有意义。这条消息含有一些经常与垃圾邮件联系在一起的单词，包括 Viagra，而这在合法的邮件中是非常罕见的，因此该消息很可能被错误地分类了。

对于类中一个或多个水平，如果一个事件从来没有发生过，就有可能出现这样的问题。例如，单词 Groceries 之前从来没有出现在垃圾邮件信息中，因此， $P(\text{垃圾邮件} | \text{groceries}) = 0$ 。

由于在贝叶斯算法中，概率值是相乘的，所以概率为 0 的值将导致该消息是垃圾邮件的后验概率值为 0，即单词 Groceries 能有效抵消或否决所有其他的证据。即使该邮件很有可能被预测为垃圾邮件，但是由于没有出现单词 Groceries 就导致了该邮件为垃圾邮件的概率为 0。

这个问题的解决涉及使用一种叫做拉普拉斯估计（Laplace estimator）的方法，该方法是以法国数学家皮埃尔 - 西蒙斯·拉普拉斯（Pierre-Simon Laplace）的名字命名的。拉普拉斯估计本质上是给频率表中的每个计数加上一个较小的数，这样就保证了每一类中每个特征发生的概率是非零的。通常情况下，拉普拉斯估计中加上的数值设定为 1，这样就保证每一类特征的组合至少在数据中出现一次。



拉普拉斯估计增加的数值可以设置为任何一个值，甚至没有必要为每一个特征设置相同的值。如果你很热衷于贝叶斯方法，你可以用拉普拉斯估计来反映一个事先假定的先验概率，这个概率是有关特征和类之间的联系。在实际应用中，在给定一个足够大的训练数据集时，这个步骤是不必要的，而且几乎总是设定其增加的数值为 1。

下面观察拉普拉斯估计如何影响我们对该消息的预测结果。取拉普拉斯值为 1，我们对每一个似然函数的分子加上 1，对分母加上分子中加上的 1 的总数。因此，我们得到垃圾邮

件的似然为：

$$(5/24) \times (11/24) \times (1/24) \times (13/24) \times (20/100) = 0.0004$$

非垃圾邮件的似然为：

$$(2/84) \times (15/84) \times (9/84) \times (24/84) \times (80/100) = 0.0001$$

这表明该消息是垃圾邮件的概率为 80%，是非垃圾邮件的概率为 20%，显然，这个结果比由单词 Groceries 单独决定结果更合理。

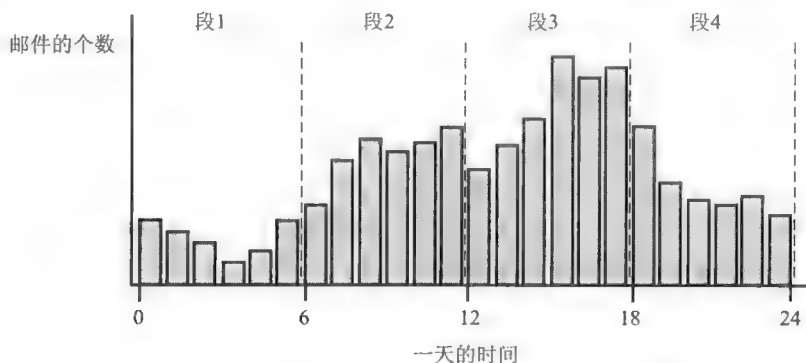
3. 在朴素贝叶斯算法中使用数值特征

由于朴素贝叶斯算法使用学习数据的频率表，所以为了创建类和特征值的组合所构成的矩阵，每个特征必须是分类变量。因为数值型特征没有类别值，所以之前的算法不能直接应用于数值型数据。然而，有一些方法可以解决这个问题。

一个简单而有效的方法就是将数值型特征值离散化，这就意味着将数值分到不同的分段 (bin) 中，这些分段就是离散化的类的水平。基于这个原因，离散化有时也称为分段。应用朴素贝叶斯算法的一个通常条件是具有大量的训练数据，这时分段的方法是理想的。

另外，还有几种不同的方法可以将数值型特征离散化。也许，最常见的方法就是探索用于自然分类的数据或者数据分布中的分割点。假设你对垃圾邮件数据集增加了一个特征，该特征就是记录了这封邮件在夜间或者白天发送的时间，它是从 0 点到午夜 24 点。

如下面的直方图所示，上述时间数据看上去可能类似于下图。在凌晨，邮件发送的频率很低；在营业期间，邮件发送活动逐渐增加，到了晚上，发送活动逐渐减少。于是，我们就可以创建 4 个活动时间的自然分段，在图中虚线的地方即为分割点。这样，数值型数据就可以划分到新的名义特征的不同水平中。这样，我们就可以应用朴素贝叶斯算法。



记住，基于数据的自然分布和一天中垃圾邮件的比例可能会随时发生变化的预感，这 4 个分段的选择某种程度上是随意的。我们可能期待垃圾邮件的发送者是在深夜时操作的；或者是在白天，当人们可能在检查邮件时，此时他们可能会进行垃圾邮件的发送。所以，为了捕捉上述的趋势，我们可以很简单地使用 3 个分段或者 12 个分段。



如果没有很明显的分割点，一种选择就是利用分位数将数值型特征离散化。你可以利用三分位数将数据划分到 3 个分段中，利用四分位数将数据划分到 4 个分段中，利用五分位数将数据划分到 5 个分段中。

有一点需要记住，将数值特征离散化总是会导致信息量的减少，因为特征的原始粒度减少为几个数目较少的类别。在处理这个问题时，重要的是取得平衡，因为太少的分段会导致重要的趋势被掩盖，而太多的分段会导致朴素贝叶斯频率表中的计数值很小。

4.2 例子——基于贝叶斯算法的手机垃圾短信过滤

随着全球手机使用量的增长，一种创造垃圾电子邮件的新途径已经为声名狼藉的营销市场开放了。这些广告商利用**短信服务 (SMS)** 文本信息，以潜在消费者为目标，给他们发送不需要的广告，即垃圾短信。这种类型的垃圾短信特别麻烦，它与垃圾邮件不同，因为许多手机用户需要为收到的每一条短信付费。研究一种可以过滤垃圾短信的分类算法，将会给移动电话供应商提供一种很有用的工具。

因为朴素贝叶斯已经成功应用于垃圾邮件的过滤，所以它很可能也可以应用于垃圾短信的过滤。然而，相对于垃圾邮件来说，垃圾短信的自动过滤有着额外的挑战。由于短信通常限制为 160 个字符，所以可以用来确定一条消息是否是垃圾消息的文本量减少了，这种限制与小的不方便的手机键盘一起，导致很多人采用短信术语简写的形式，这进一步模糊了合法消息和垃圾消息的界限。让我们看一看一个简单的朴素贝叶斯分类器如何处理好这些问题带来的挑战。

4.2.1 第 1 步——收集数据

为了扩展朴素贝叶斯分类器，我们将使用从网站 <http://www.dt.fee.unicamp.br/~tiago/sms-spamcollection/> 收集的垃圾短信改编的数据。



要了解更多关于垃圾短信的收集，可以参考作者发表的文章：On the Validity of a New SMS Spam Collection by J.M. Gómez Hidalgo, T.A. Almeida and A. Yamakami in Proceedings of the 11th IEEE International Conference on Machine Learning and Applications (2012)。

该数据集包含短信的文本信息，而且带有表明该短信是否为垃圾短信的标签。垃圾短信标记为 spam，而非垃圾短信标记为 ham。在下面的例子中有一些关于垃圾短信和非垃圾短信的例子。

下面是一些非垃圾短信的例子：

Better. Made up for Friday and stuffed myself like a pig yesterday. Now I feel bleh. But at least its not writhing pain kind of bleh.

If he started searching he will get job in few days. He have great potential and talent.

I got another job! The one at the hospital doing data analysis or something, starts on monday!
Not sure when my thesis will got finished

下面是一些垃圾短信的例子：

Congratulations ur awarded 500 of CD vouchers or 125gift guaranteed & Free entry 2 100 wkly draw txt MUSIC to 87066

December only! Had your mobile 11mths+? You are entitled to update to the latest colour camera mobile for Free! Call The Mobile Update Co FREE on 08002986906

Valentines Day Special! Win over £1000 in our quiz and take your partner on the trip of a lifetime! Send GO to 83600 now. 150p/msg rcvd.

看到前面的短信示例，你是否注意到垃圾短信的显著特点吗？一方面，一个显著特点是 3 条垃圾短信中有 2 条短信使用了单词 free，但该单词没有出现在任何一条非垃圾短信中。另一方面，与垃圾短信相比，有 2 条非垃圾短信引用了一周中具体的某一天，而垃圾短信中没有一条引用。

我们的朴素贝叶斯分类器将利用词频中这种模式的优势来确定短信消息是更像垃圾短信还是非垃圾短信。尽管可以想象单词“free”可以出现在非垃圾短信中，但是一条合法的短信很有可能会根据上下文提供额外的单词信息。例如，一条非垃圾短信可能会这样陈述“are you free on Sunday?”；而一条垃圾短信可能会使用这样的短语“free ringtones”。朴素贝叶斯分类器将根据短信中所有单词提供的证据计算垃圾短信和非垃圾短信的概率。

4.2.2 第 2 步——探索和准备数据

构建分类器的第一步涉及原始数据的处理与分析，文本数据的准备具有挑战性，因为将词和句子转化成计算机能够理解的形式是很必要的。我们将把数据转化成一种称为词袋 (bag-of-words) 的表示方法，这种表示方法忽略了单词出现的顺序，只是简单地提供一个变量用来表示单词是否会出现。



为了使数据可以在 R 中方便地应用，这里所使用的数据已经对原始数据进行修正。如果你想跟着一起运行这个例子，可以从网站下载 sms_spam.csv 文件，并将其保存到 R 的工作目录下。

我们首先使用 read.csv() 函数导入上述 CSV 数据，将其保存到以 sms_raw 命名的数据文件中。

```
> sms_raw <- read.csv("sms_spam.csv", stringsAsFactors = FALSE)
```

使用数据结构探索函数 str()，可以看到 sms_raw 数据文件包含了 5559 条短信，每条短信都有两个特征：type 和 text。将 SMS 的特征 type 编码为 ham 或者 spam，而变

量 `text` 存储整个 SMS 短信文本。

```
> str(sms_raw)
'data.frame': 5559 obs. of 2 variables:
 $ type: chr "ham" "ham" "ham" "spam" ...
 $ text: chr "Hope you are having a good week. Just checking in" "K..
give back my thanks." "Am also doing in cbe only. But have to pay."
"complimentary 4 STAR Ibiza Holiday or £10,000 cash needs your URGENT
collection. 09066364349 NOW from Landline not to lose out"| __truncated__
...
```

当前的变量 `type` 是一个字符串向量。由于它是一个分类变量，所以将其转换成一个因子将会更好，如下面的代码所示：

```
> sms_raw$type <- factor(sms_raw$type)
```

用函数 `str()` 和 `table()` 研究变量 `type`，可以看到该变量已经被很好地重新编码为一个因子。此外，可以看到数据中有 747 条，（即大约 13%）短信被标记为 `spam`，其余的短信被标记为 `ham`。

```
> str(sms_raw$type)
Factor w/ 2 levels "ham","spam": 1 1 1 2 2 1 1 1 2 1 ...
> table(sms_raw$type)
ham spam
4812 747
```

现在，我们先不研究变量 `text`。你将在下一节中学习，因为处理原始短信需要使用一套新的专门用于处理文本数据的功能强大的工具。

4.2.3 数据准备——处理和分析文本数据

短信就是由词、空格、数字和标点符号组成的文本字符串。处理这种类型的复杂数据需要大量的思考和工作，一方面需要考虑如何去除数字和标点符号，如何处理没有意义的单词，如 `and`、`but` 和 `or` 等，以及如何将句子分解成单个的单词。幸运的是，R 社区的成员已经在文本挖掘添加包 `tm` 中提供了这些功能。



添加包 `tm` 最初是由维也纳财经大学 (Vienna University of Economics and Business) 的 Ingo Feinerer 作为一个论文项目创建的。如果想了解这个添加包的更多知识，你可以访问网站 <http://tm.r-forge.r-project.org/>。

可以通过命令 `install.packages("tm")` 安装 `tm` 文本挖掘添加包，并应用命令 `library(tm)` 进行加载。

处理文本数据的第一步涉及创建一个语料库，即一个文本文件的集合。在我们的例子中，一个文本文件就是指一条短信，我们通过下面的命令建立一个包含训练数据中短信的语料库。

```
> sms_corpus <- Corpus(VectorSource(sms_raw$text))
```

这条命令使用了两个函数。首先，函数 `Corpus()` 创建了一个 R 对象来存储文本文档。这个函数通过一个参数来指定所加载的文本文档的格式。因为我们已经把短信读入 R 并存储

在一个向量中，所以我们用函数 `VectorSource()` 来指示函数 `Corpus()` 使用向量 `sms_train$text` 的信息。函数 `Corpus()` 将结果存储在一个名为 `sms_corpus` 的对象中。



函数 `Corpus()` 非常灵活，可以读入很多不同来源的文档，比如 PDF 和 Microsoft Word 文档。要了解更多信息，可以通过命令 `print(vignette("tm"))` 查看 `tm` 添加包中 Data Import（数据导入）部分的简介。

如果我们用函数 `print()` 输出我们刚刚创建的语料库，我们将会看到该语料库包含了训练数据中 5559 条短信的每一条短信。

```
> print(sms_corpus)
A corpus with 5559 text documents
```

如果要查看语料库的内容，可以使用函数 `inspect()`。将该函数与访问向量的方法结合在一起，可以查看具体的短信内容。下面的命令就是查看第一、二、三条短信的具体内容：

```
> inspect(sms_corpus[1:3])
[[1]]
Hope you are having a good week. Just checking in
[[2]]
K..give back my thanks.
[[3]]
Am also doing in cbe only. But have to pay.
```

语料库现在包含 5559 条短信的原始文本内容。在将文本内容分解成单词之前，我们需要进行一些清理步骤以去除标点符号和可能会影响结果的其他字符。例如，我们将把单词 `hello!`、`HELLO……` 和 `Hello` 都作为单词 `hello` 的实例。

函数 `tm_map()` 提供了一种用来转换（即映射）`tm` 语料库的方法。我们将使用一系列转换函数来清理我们的语料库，并将结果保存为一个叫做 `corpus_clean` 的新对象。

首先，我们将把所有短信的字母变成小写字母，并去除所有的数字：

```
> corpus_clean <- tm_map(sms_corpus, tolower)
> corpus_clean <- tm_map(corpus_clean, removeNumbers)
```

在分析文本数据时，一个常见的做法就是去除填充词，比如 `to`、`and`、`but` 和 `or`，这些词称为停用词（stop word）。我们将使用 `tm` 添加包中提供的函数 `stopwords()`，而不是我们自己定义一个停用词列表，这个函数包含了一组大量的停用词。如果想知道它包含的所有停用词，在命令行输入 `stopwords()` 即可。正如前面所做的，我们将通过使用函数 `tm_map()` 来剔除数据中的停用词。

```
> corpus_clean <- tm_map(corpus_clean, removeWords, stopwords())
```

同样，我们也可以去除标点符号：

```
> corpus_clean <- tm_map(corpus_clean, removePunctuation)
```

既然我们已经去除了数字、停用词和标点符号，那么文本信息中这些字符曾经所在的地方就变成了空格。最后一步就是去除额外的空格，只在词与词之间留下一个空格。

```
> corpus_clean <- tm_map(corpus_clean, stripWhitespace)
```

下表显示了短信语料库中前3条短信在清理前后的对比。短信消息已经被限制得只剩下最有意义的词，标点符号和大小写都已经被清理。

短信清理之前	短信清理之后
<pre>> inspect(sms_corpus[1:3])</pre>	<pre>> inspect(corpus_clean[1:3])</pre>
[[1]]	[[1]]
Hope you are having a good week. Just checking in	hope good week just checking
[[2]]	[[2]]
K..give back my thanks.	kgive back thanks
[[3]]	[[3]]
Am also doing in cbe only. But have to pay.	also cbe pay

既然以我们想要的方式处理了数据，那么最后的步骤就是通过一个所谓的**标记化**过程将消息分解成由单个单词组成的组。一个记号（token）就是一个文本字符串的单个元素，在这种情况下，本例中的记号就是单词。



这里的例子是在 Microsoft Windows 7 环境下运行的，用的是 0.5 ~ 9.1 版本的 tm 添加包，R 的版本是 2.15.3。如果你使用其他的 R 语言版本或者另一个平台，由于这些项目在不断变化，所以所得的结果可能会略有同。

正如你预想的那样，tm 添加包提供了标记短信语料库的功能。函数 `DocumentTermMatrix()` 将一个语料库作为输入，并创建一个称为**稀疏矩阵**的数据结构，其中矩阵的行表示文档（即短信），矩阵的列表示单词。矩阵中的每个单元存储一个数字，它代表由列标识的单词出现在由行所标识的文档中的次数。下面的截图显示了短信语料库的文档-单词（稀疏矩阵）矩阵的一小部分，而作为完整的稀疏矩阵则有 5559 行和超过 7000 列。

	A	B	C	D	E	F	G
		balloon	balls	bam	bambling	band	bandages
1		0	0	0	0	0	0
2		0	0	0	0	0	0
3		0	0	0	0	0	0
4		0	0	0	0	0	0
5		0	0	0	0	0	0

事实上，上表中的每一个格子中的 0 代表它们所在列的顶部所列出的单词没有出现在语料库的前 5 条短信中的任意一条中，这突出表明这个数据结构被称为稀疏矩阵的原因，即在该矩阵中，绝大多数元素是以 0 来填充的。尽管每个消息包含了一些单词，但是任意具体的单词出现在给定的一条消息中概率都是很小的。

给定 tm 语料库，创建一个稀疏矩阵要用到下述命令：

```
> sms_dtm <- DocumentTermMatrix(corpus_clean)
```

这条命令将语料库标记化，并返回一个名为 `sms_dtm` 的稀疏矩阵。从这里，我们就可

以对包括词频在内的信息进行分析。

1. 数据准备——建立训练数据集和测试数据集

由于已经为分析准备好了数据，所以现在我们需要将数据分成训练数据集和测试数据集，从而可以把垃圾短信分类器应用到之前没有学习过的数据上，并据此对分类器的性能进行评估。我们将数据分成两部分：75% 的训练数据和 25% 的测试数据。因为短信的排序是随机的，所以我们可以简单地取前 4169 条短信用于训练，剩下的 1390 条短信用于测试。

我们将通过分解原始数据框开始：

```
> sms_raw_train <- sms_raw[1:4169, ]
> sms_raw_test  <- sms_raw[4170:5559, ]
```

然后输出文档-单词矩阵：

```
> sms_dtm_train <- sms_dtm[1:4169, ]
> sms_dtm_test  <- sms_dtm[4170:5559, ]
```

最后，得到语料库：

```
> sms_corpus_train <- corpus_clean[1:4169]
> sms_corpus_test  <- corpus_clean[4170:5559]
```

为了确认上述子集是一组完整的短信数据的代表，可以通过比较垃圾短信在训练数据和测试数据中所占的比例：

```
> prop.table(table(sms_raw_train$type))
      ham      spam
0.8647158 0.1352842
> prop.table(table(sms_raw_test$type))
      ham      spam
0.8683453 0.1316547
```

无论是训练数据集和测试数据集，它们都包含约 13% 的垃圾短信，这表明垃圾短信被平均分配在这两个数据集中。

2. 可视化文本数据——词云

词云是一种可视化地描绘单词出现在文本数据中频率的方式。词云是由随机分布在词云图中的单词构成的，经常出现在文本中的单词会以较大的字体呈现，而不太常见的单词将会以较小的字体呈现。最近，这种类型的图已经变得越来越流行，因为它提供了一种观察社交媒体网站上热门话题的方式。

wordcloud 添加包提供了一个简单的 R 函数来创建这种类型的图形，我们将应用这个函数使短信中单词类型可视化。比较垃圾短信和非垃圾短信的词云将有助于我们了解朴素贝叶斯短信过滤器是否有可能成功。如果还没有安装 wordcloud 添加包，你需要在 R 命令行输入命令 `install.packages("wordcloud")` 来安装这个添加包，并输入 `library(wordcloud)` 来加载它。



wordcloud 添加包是由加州大学洛杉矶分校的专业统计学家 Ian Fellows 编写的。想要了解关于这个添加包的更多信息，可以访问网站：<http://cran.r-project.org/web/packages/wordcloud/index.html>。

可以从 tm 语料库对象直接创建词云，命令如下所示：

```
> wordcloud(sms_corpus_train, min.freq = 40, random.order = FALSE)
```

该命令将从 sms_corpus_train 语料库创建一个词云。由于我们设置 random.order = FALSE，所以该词云将以非随机的顺序排列，而且出现频率越高的单词越靠近中心。如果我们没有设置 random.order，该词云将会以默认的随机方式排列。参数 min.freq 用来指定显示在词云中的单词必须满足在语料库中出现的最小次数。通用规则是，开始时设置参数 min.freq 的值为语料库中文档总数目的 10%。这里，文档总数目的 10% 大约为 40。因此，词云中的单词至少在 40 条短信中出现过。



你可能会得到一条警告消息指出 R 无法将所有的单词显示在图中。如果这样，试着将参数 min.freq 的值向上调整，从而减少词云中单词的数量。另外，使用参数 scale 改变字体的大小也很有用。

该词云的结果如下图所示。

另一个有趣的可视化涉及垃圾短信和非垃圾短信词云的比较。由于我们没有对垃圾短信和非垃圾短信分别建立语料库，所以这时应该应用函数 wordcloud() 一个非常有用的功能。给定原始文本，在建立语料库和显示词云之前，它会自动应用文本转换过程。

我们可以根据短信类型，通过 R 函数 subset() 来获取其中一种类型的短信数据，即抽取 sms_raw_train 的一个子集。首先，我们将创建 type 等于 spam 的子集：

```
> spam <- subset(sms_raw_train, type == "spam")
```

其次，获取子集 ham：

```
> ham <- subset(sms_raw_train, type == "ham")
```



要注意双等号，与很多程序语言一样，R 使用 “==” 表示相等。如果你一不小心使用了一个等号，你将会得到一个比你预期要大得多的子集。

现在，我们有两个短信数据框：spam（垃圾短信）和 ham（非垃圾短信），每一个都带有包含原始文本字符串的文本特征。创建词云就像之前一样简单。这一次，我们将使用参数 max.words，它用来显示两个集合的任何一个集合中最常见的 40 个单词，而且参数 scale 允许调整词云中单词的最大字体和最小字体。你可以自由调整这些参数直到你认为合适。如


```
> sms_train <- DocumentTermMatrix(sms_corpus_train,
  list(dictionary = sms_dict))
> sms_test <- DocumentTermMatrix(sms_corpus_test,
  list(dictionary = sms_dict))
```

现在，训练数据和测试数据包含了大约 1200 个特征，只对应于至少出现在 5 条短信中的单词。

朴素贝叶斯分类器通常是训练具有明确特征的数据，这就带来了一个问题，因为稀疏矩阵中的元素表示一个单词出现在一条消息中的次数。于是，我们需要将其改变成因子变量，根据单词是否出现，简单地表示成 yes 或者 no。

下面的代码定义了一个函数 `convert_counts()`，它将计数转换成因子：

```
> convert_counts <- function(x) {
  x <- ifelse(x > 0, 1, 0)
  x <- factor(x, levels = c(0, 1), labels = c("No", "Yes"))
  return(x)
}
```

到现在为止，上面函数中的一些模块看上去应该很熟悉了。其中，第一行用来定义函数，语句 `ifelse(x > 0, 1, 0)` 将改变 `x` 中的值，如果该值大于 0，则它会被 1 代替，否则，它仍然为 0。命令 `factor` 简单地将值 0 和 1 转化为用 no 和 yes 所标记的因子。最后，返回变换后的向量 `x`。

现在，我们只需要将 `convert_counts` 应用于稀疏矩阵的每一列。你也许能够猜到 R 中的函数可以做到这点，该函数已经在前面提到过。这个函数就是 `apply()` 函数。



`apply()` 函数是包括 `lapply()` 和 `sapply()` 函数在内的函数族中的一员，这些函数可以作用于 R 数据结构中的每一个元素，而且这些函数是 R 语言的关键词之一。经验丰富的 R 程序开发人员使用这些函数而不使用循环（比如你在其他程序语言中使用的 `for` 和 `while` 语句等），因为这些函数代码更具有可读性而且有时会更高效。

函数 `apply()` 允许一个函数作用于一个矩阵的每一行或者每一列，它使用参数 `MARGIN` 来指定作用的对象是矩阵的行或者列。在本例中，我们感兴趣的是矩阵的列，所以我们令 `MARGIN = 2`（`MARGIN = 1` 表示的是行）。用来转换训练矩阵和测试矩阵的完整命令如下所示：

```
> sms_train <- apply(sms_train, MARGIN = 2, convert_counts)
> sms_test <- apply(sms_test, MARGIN = 2, convert_counts)
```

结果将是两个矩阵，每个矩阵都带有因子类型的列，用 Yes 和 No 来表示每一列的单词是否出现在构成行的信息中。

4.2.4 第 3 步——基于数据训练模型

因为我们已经将原始短信转换成了可以用一个统计模型代表的形式，所以此时是应用朴

素贝叶斯算法的时候了。该算法将根据单词的存在与否来估计一条给定的短信是垃圾短信的概率。

我们采用 e1071 添加包中的朴素贝叶斯算法实现。这个添加包是由维也纳理工大学 (Vienna University of Technology, TU Wien) 统计系开发的, 它包含了用于机器学习的多种函数。如果你还没有安装这个添加包, 在继续之前, 需要使用命令 `install.packages("e1071")` 和 `library(e1071)` 来准备好这个包。



许多机器学习方法不只是通过一个 R 添加包实现的, 朴素贝叶斯也不例外。

另一种经常引用的朴素贝叶斯函数是 klaR 添加包中的 `NaiveBayes()` 函数, 它几乎等同于本文中所描述的函数。你可以自由选择你喜欢的函数。

与前面章节中我们用于分类的 KNN 算法不同, 训练一个朴素贝叶斯分类器和使用朴素贝叶斯进行分类是发生在不同的阶段的。尽管如此, 分类相当简单, 如下表所示。

朴素贝叶斯分类语法
应用 e1071 添加包中的函数 <code>naiveBayes()</code>
<p>创建分类器:</p> <pre>m <- naiveBayes(train, class, laplace = 0)</pre> <ul style="list-style-type: none"> • <code>train</code>: 数据框或者包含训练数据的矩阵 • <code>class</code>: 包含训练数据每一行的分类的一个因子向量 • <code>laplace</code>: 控制拉普拉斯估计的一个数值 (默认为 0) <p>该函数返回一个朴素贝叶斯模型对象, 该对象能够用于预测。</p> <p>进行预测:</p> <pre>p <- predict(m, test, type = "class")</pre> <ul style="list-style-type: none"> • <code>m</code>: 由函数 <code>naiveBayes()</code> 训练的一个模型 • <code>test</code>: 数据框或者包含测试数据的矩阵, 包含与用来建立分类器的训练数据相同的特征 • <code>type</code>: 值为 "class" 或者 "raw", 标识预测是最可能的类别值或者原始的预测概率 <p>该函数将返回一个向量, 根据参数 <code>type</code> 的值, 该向量含有预测的类别值或者原始预测的概率值。</p> <p>例子:</p> <pre>sms_classifier <- naiveBayes(sms_train, sms_type) sms_predictions <- predict(sms_classifier, sms_test)</pre>

为了基于 `sms_train` 矩阵建立模型, 我们将使用如下的命令:

```
> sms_classifier <- naiveBayes(sms_train, sms_raw_train$type)
```

`sms_classifier` 变量现在包含一个可以用于预测的 `naiveBayes` 分类器对象。

4.2.5 第 4 步——评估模型的性能

为了评估短信分类器, 我们需要基于测试数据中未知的短信来检验分类器的预测值。回想一下, 未知的短信特征存储在一个名为 `sms_test` 的矩阵中, 而分类标签 `spam` 和 `ham` 存储在 `sms_raw_test` 数据框中一个名为 `type` 的向量中。我们把已经训练过的分类器命

名为 `sms_classifier`，我们将用它来产生预测值，并将预测值与真实值相比较。

我们用函数 `predict()` 进行预测，并将这些预测值存储在一个名为 `sms_test_pred` 的向量中：

```
> sms_test_pred <- predict(sms_classifier, sms_test)
```

为了比较预测值和真实值，我们将使用 `gmodels` 添加包中的函数 `CrossTable()`，在之前我们已经使用过这个函数。这一次，我们将增加一些额外的参数来消除不必要的元素的比例，并使用参数 `dnn`（维度名称）来重新标记行和列，如下面的代码所示：

```
> library(gmodels)
> CrossTable(sms_test_pred, sms_raw_test$type,
  prop.chisq = FALSE, prop.t = FALSE,
  dnn = c('predicted', 'actual'))
```

这就产生了下面的表格：

Total Observations in Table: 1390

predicted	actual		row Total
	ham	spam	
ham	1203 0.997	32 0.175	1235
spam	4 0.003	151 0.825	155
Column Total	1207 0.868	183 0.132	1390

从表中可以看出，1207 条非垃圾短信中有 4 条短信被错误地归为垃圾短信，比例为 0.3%，而 183 条垃圾短信中有 32 条短信被错误地归为非垃圾短信，比例为 17.5%。考虑到在这个案例中，我们几乎没有做什么工作，具有这种水平的表现是相当好的。另外，本案例的研究也说明了为什么说朴素贝叶斯算法是用于文本分类的一种标准算法，而且朴素贝叶斯方法可以直接拿来使用，执行的效果也是出奇地好。

另一方面，被错误地归为垃圾短信的 4 条短信可能会为过滤算法的部署带来显著的问题。如果过滤器导致某人错过一条重要的约会或者紧急情况的短信，那么他们会很快放弃这种产品，因此我们需要研究这些错误分类的短信，看看是哪里出了问题。

4.2.6 第 5 步——提升模型的性能

你可能已经注意到，在训练模型时，我们并没有设置一个值来用于拉普拉斯估计。毫无疑问，这样就允许在分类的过程中单词可能出现在 0 条垃圾短信或者 0 条非垃圾短信中。比如，虽然单词“ringtone”只出现在训练数据的垃圾短信中，但这并不意味着每一个含有单词“ringtone”的短信就应该被归为垃圾短信。

我们会像之前一样建立朴素贝叶斯模型，但这一次我们设置 `laplace = 1`：

```
> sms_classifier2 <- naiveBayes(sms_train, sms_raw_train$type,
  laplace = 1)
```

接下来，我们将进行预测：

```
> sms_test_pred2 <- predict(sms_classifier2, sms_test)
```

最后，我们将使用交叉表来比较预测的分类和真实的分类情况：

```
> CrossTable(sms_test_pred2, sms_raw_test$type,
  prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
  dnn = c('predicted', 'actual'))
```

该结果如下表所示。

Total observations in Table: 1390

predicted	actual		Row Total
	ham	spam	
ham	1204 0.998	31 0.169	1235
spam	3 0.002	152 0.831	155
Column Total	1207 0.868	183 0.132	1390

我们不仅将错误地归为垃圾短信的非垃圾短信的数量由 4 减少到 3，而且我们也将错误地归为非垃圾短信的垃圾短信数量由 32 减少到 31。虽然这看上去是一个很小的改进，但我们必须意识到如果我们对垃圾短信的过滤过于激进，那么重要信息被遗漏的可能性是很大的。

4.3 总结

本章中，我们学习了如何使用朴素贝叶斯进行分类。该算法构建了概率表用来估计新案例属于不同类别的似然。概率是通过一个称为贝叶斯定理的公式来计算的，它表明相关事件是如何相关的。尽管贝叶斯公式的计算是很复杂的过程，但是应用一个事件相互独立的“简单”假设后，就得到一个可用于巨大数据集的简化贝叶斯算法。

朴素贝叶斯分类器通常用于文本分类。为了说明其有效性，我们采用朴素贝叶斯进行了一个关于过滤垃圾短信的分类任务。需要专门的 R 添加包用于准备需要分析的文本数据、预处理文本以及文本的可视化。最终，该模型能够将大约 98% 的短信正确地分成垃圾短信和非垃圾短信。

在第 5 章中，我们将研究另两种机器学习方法，每种方法都通过将数据划分到具有相似值的组中进行分类。

分而治之——应用决策树和规则进行分类

为了做出一个艰难的决定，有些人会列出每一种可能性的利与弊来权衡他们的选择。假设一个求职者正在几份工作机会之间做抉择，而这些工作有的离家近，有的离家远，且有着不同水平的薪酬和福利，他可能会创建一张带有每一个职位特征的列表，然后根据这些特征，创立规则来排除一些选择。比如，“如果我上下班时间超过一个小时，那么我会不高兴”，或者，“如果我挣的钱少于 5 万美元，那么我将不能够支撑我的家庭”。预测未来幸福的困难决定可以简化为一系列小的越来越具体的选择。

本章介绍两种机器学习方法——决策树和规则学习。这两种方法应用一个类似的策略：通过将数据集划分成较小的子集来确定用于预测的方式，这些知识之后将以逻辑结构的形式呈现，不需要任何统计知识就可以理解。这方面使得这些模型对改进企业战略和业务流程特别有用。

到本章结束时，你将学习下列知识：

- 每一种方法是采用什么策略将数据划分成令人感兴趣的类别的。
- 决策树和规则分类的几种实现方法，包括 C5.0 算法、1R 算法和 RIPPER 算法
- 如何使用这些算法进行现实世界的分类任务，比如，确定高风险的银行贷款、识别有毒的蘑菇。

我们首先研究决策树，其次研究规则分类，最后我们总结本章所学到的内容并预览后面的章节。在后面的章节中，我们将讨论以决策树和规则学习为基础，用于其他高级机器学习技术的方法。

5.1 理解决策树

正如你可能会从名字知道，决策树学习算法以**树形结构**建立模型。类似于流程图，该模

型本身包含一系列逻辑决策，带有表明根据某一属性做出决定的**决策节点**。从这些决策节点引出的分枝表示可做出的选择，决策树由叶节点（leaf notes，也称为终端节点）终止，叶节点表示遵循决策组合的结果。

数据的分类从**根节点**开始，根据特征值遍历树上的各个决策节点。数据采用的是一个漏斗形的路径，它将每一条记录汇集到一个叶节点，在叶节点为该记录分配一个预测类。

因为决策树本质上是一个流程图，所以决策树特别适合应用于由于法律因素需要透明化的分类机制以及为了便于决策需要共享成果的分类。一些潜在的用途包括：

- ❑ 信用评估模型中，其中导致申请被拒绝的准则必须明确规定
- ❑ 客户流失或者客户满意度的市场调查将与管理机构或者广告公司共享
- ❑ 基于实验测量、症状或者疾病进展率的医疗条件诊断

虽然前面的应用都说明了用于展现决策过程的决策树的价值，但这并不说明决策树的效用到此为止。事实上，决策树可能是最广泛使用的机器学习技术之一，它几乎可以用于任何类型的数据建模，并且具有无与伦比的性能。

尽管决策树的应用性很广，但值得注意的是，在一些案例中，决策树可能不是一个理想的选择。这样的案例可能是下述情况下的分类任务，此时数据有大量的多层次的名义特征或者数据有大量的数值特征，这些案例可能生成数量庞大的决策和一个过于复杂的决策树。

5.1.1 分而治之

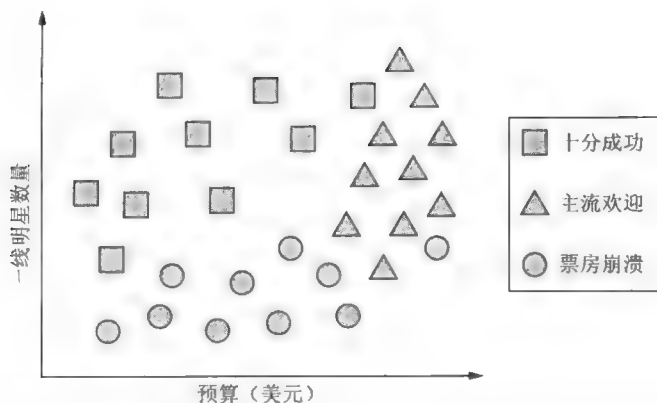
决策树的建立使用一种称为**递归划分 (recursive partitioning)**的探索法。这种方法通常称为**分而治之 (Divide and Conquer)**，因为它利用特征的值将数据分解成具有相似类的较小的子集。

从代表整个数据集的根节点开始，该算法选择最能预测目标类的特征，然后，这些案例将被划分到这一特征的不同值的组中，这一决定形成了第一组树枝。该算法继续分而治之其他节点，每次选择最佳的候选特征，直到达到停止的标准。如果一个节点停止，它可能具有下列情况：

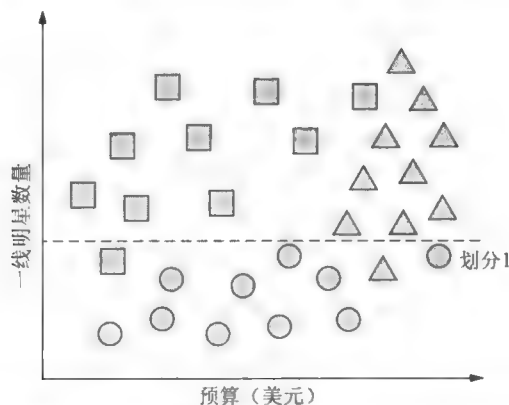
- ❑ 节点上所有（几乎所有）的案例都属于同一类。
- ❑ 没有剩余的特征来分辨案例之间的区别。
- ❑ 决策树已经到达预先定义的大小限制。

为了说明决策树的建立过程，我们来考虑一个简单的例子。想象你正在一家好莱坞电影制片厂工作，你的办公桌上堆满了剧本，你决定研究一个决策树算法来预测一部有潜力的电影是否会落入受主流欢迎（mainstream hit）、批评家的选择（critic's choice）和票房崩溃（box office bust）这三大类中，而不是从头到尾把每个剧本读一遍。

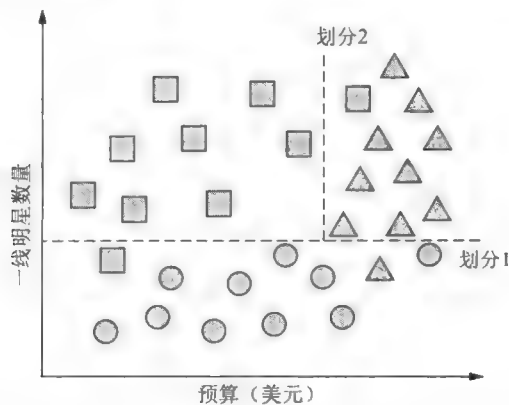
为了收集数据来建立你的模型，你转向工作室的档案去研究过去 10 年电影发行的情况。在查看了 30 个不同电影剧本的数据后，一个模型出现了，在电影的拍摄预算中一线名人排队等候主角角色的数量与电影成功的类别之间似乎有一种关系。该数据的散点图可能类似于下面这幅图：



为了使用该数据建立一个简单的决策树，我们可以应用一个分而治之策略。首先，我们将表示名人数量的特征进行划分，将电影划分成有少数一线明星和有很多一线明星的两组：



其次，在有很多名人的这组电影中，我们可以根据电影预算的高低做另一个划分：



此时，我们已经将数据划分成3组。图左上角的这组完全由广受好评的电影组成，这一组是根据有相当多的名人和相对较低的预算划分出来的；图右上角的这组是由大多数票房很

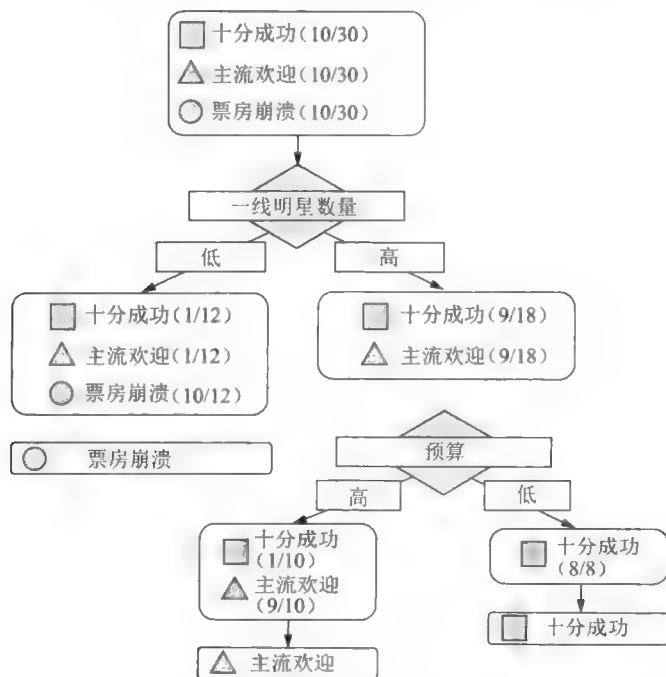
高的电影组成的，有很高的预算和相当多的名人；最后一组，几乎没有明星的力量，预算有高有低，也包括了比较失败的电影。

如果需要，我们可以继续划分数据，根据越来越具体的预算范围和名人数量，直到每个错误的分类值都分配到自己的分区（或许是很微小的分区）。由于数据可以继续划分，直到在一个分区内的特征没有区别，所以决策树容易对训练数据进行过度拟合，给出过于具体细节的决策。我们将在这里停止分类算法以避免这种情况发生，因为每组中超过 80% 的案例来自于同一个类。



你可能已经注意到，对角线可能更加清晰地划分数据。这是决策树算法使用**轴平行分割**来表现知识的一个局限性。每一分割一次只考虑一个特征的事实，预防了决策树会由更复杂的决策形成，比如，“如果名人的数量远大于所估计的预算，那么这部电影将会是一部很成功（critical success 类）的电影。”

我们用于预测电影未来成功的模型可以用一个简单的决策树来表示，如下图所示。为了评估一个剧本，可以依照每一个决策分枝，直到预测出它是成功的或者失败的。于是，在任何时候，你都可以对积压的剧本进行分类，然后去做更重要的工作，比如写一份获奖感言。



由于现实世界的数据包含的不仅仅是两个特征，所以决策树很快就会变得比上图复杂，会有更多的节点、分枝和叶子。在下一节中，你将学习一个自动建立决策树模型的流行算法。

5.1.2 C5.0 决策树算法

决策树有很多算法可以实现，但 C5.0 算法是最知名的决策树算法之一。该算法是由计算机科学家 J.Ross Quinlan 为改进他之前的算法 C4.5 开发的新版本，C4.5 本身就是对他的 ID3 算法（Iterative Dichotomiser 3，迭代二叉树 3 代）的一个改进。尽管 Quinlan 将 C5.0 算法销售给商业用户（详情可见网站 <http://www.rulequest.com/>），但是该算法的一个单线程版本的源代码是公开的，因此可以编写成程序，比如 R 中就有该程序



对于更混乱的问题，一个基于 Java 的开源流行算法名为 J48，可以代替 C4.5 算法，该算法包含在 RWeka 包中。因为 C5.0、C4.5 和 J48 算法之间的差异是很小的，所以本章的原则适用于这 3 种算法中的任意一种，而且这 3 种算法应该认为是同义的。

C5.0 算法已经成为生成决策树的行业标准，因为它确实适用于大多数类型的问题，而且可以直接使用。与其他先进的机器学习模型（比如第 7 章所描述的黑箱方法——神经网络和支持向量机）相比，通过 C5.0 算法建立的决策树一般都表现得与其他先进的模型几乎一样好，而且更容易理解和易于部署。此外，该算法的缺点相对来说是较轻微的，而且在很大程度上都可以避免，如下表所示。

优点	缺点
<ul style="list-style-type: none">• 一个适用于大多数问题的通用分类器• 高度自动化的学习过程，可以处理数值型数据、名义特征以及缺失数据• 只使用最重要的特征• 可以用于只有相对较少训练案例的数据或者有相当多训练案例的数据• 没有数学背景也可以解释一个模型的结果（对于比较小的树）• 比其他复杂的模型更有效	<ul style="list-style-type: none">• 决策树模型在根据具有大量水平的特征进行划分时往往是有偏的• 很容易过度拟合或者不能充分拟合模型• 因为依赖于轴平行分割，所以在对一些关系建立模型时会有困难• 训练数据中的小变化可能导致决策逻辑的较大变化• 大的决策树可能很难理解，给出的决策可能看起来会违反直觉

在本章的前面，我们用一个简单的例子说明了决策树模型的数据如何使用分而治之策略。让我们更详细地探讨这种方法，研究该探索法在实际中是如何运作的。

1. 选择最佳的分割

决策树面临的第一个挑战就是需要确定根据哪个特征进行分割。在前面的例子中，我们以这样一种方式来寻找分割数据的特征值，即分区中主要包含来源于一个单一类的案例。如果一组数据中只包含一个单一的类，那么这些类被认为是**纯的**。有许多不同的度量纯度的方法，它们可以用来确定分割的标准。

C5.0 算法使用**熵**度量纯度。样本数据的熵表示分类值如何混杂在一起，最小值 0 表示的样本是完全同质的，而 1 表示样本凌乱的最大数量。熵的定义具体如下：

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

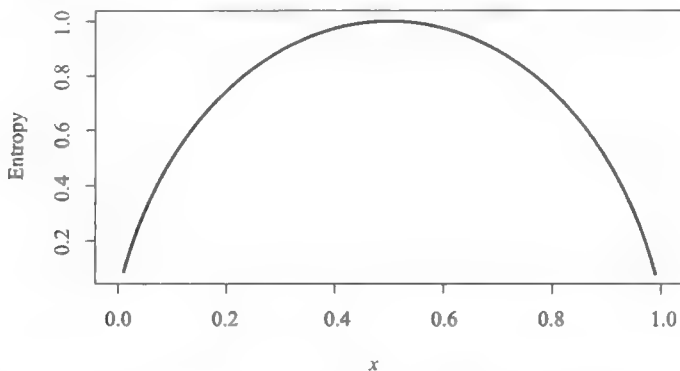
在熵的公式中，对于给定的数据分割 (S)，常数 c 代表类的水平数， p_i 代表落入类的水平 i 中的特征值的比例。例如，假设我们有一个两个类的数据分割：红 (60%) 和白 (40%)，我们可以计算该数据分割的熵，如下所示：

```
> -0.60 * log2(0.60) - 0.40 * log2(0.40)
[1] 0.9709506
```

我们可以考察所有可能的两个类划分的熵。如果我们知道在一个类中案例的比例为 x ，那么在另一个类中的比例就是 $1-x$ 。使用函数 `curve()`，我们就可以绘制关于 x 的所有可能值的熵的图形：

```
> curve(-x * log2(x) - (1 - x) * log2(1 - x),
        col="red", xlab = "x", ylab = "Entropy", lwd=4)
```

结果如下图所示。



如上图所示，熵的峰值在 $x=0.5$ 时取到，一个 50-50 分割导致最大熵值。当一个类相对于其他类越来越占据主导地位时，熵值会逐渐减少到 0。

给定这种度量纯度的方式，决策树算法仍然必须决定根据哪个特征进行分割。对于这一点，决策树算法使用熵值来计算由每一个可能特征的分割所引起的同质性（均匀性）变化，该计算称为**信息增益**。对于特征 F ，信息增益的计算方法是分割前的数据分区 (S_1) 的熵值减去由分割产生的数据分区 (S_2) 的熵值，即

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

复杂之处在于，一次分割后，数据被划分到多个分区中，因此计算 $\text{Entropy}(S_2)$ 的函数需要考虑所有分区熵值的总和。这可以通过记录落入每一分区的比例来计算每一分区的权重，可以用如下的公式来表示：

$$\text{Entropy}(S) = \sum_{i=1}^n w_i \text{Entropy}(P_i)$$

简单来说，从一个分割得到的总熵就是根据案例落入分区中的比例 w_i 加权的 n 个分区的熵值的总和。

信息增益越高，根据某一特征分割后创建的分组越均衡，如果信息增益为零，那么根据该特征进行分割后的熵值就不会减少。另一方面，最大信息增益等于分割前的熵值，这意味着分割后熵值为零，即决策结果是在完全同质的分组中。

前面的公式假定是名义特征，但对于数值特征的分割，决策树同样可以使用信息增益。一个通常的做法就是测试不同的分割，根据比阈值大还是比阈值小，将数值划分到不同的组中，这将数值特征压缩到一个两水平的分类特征，而且信息增益可以很容易计算，选择产生最大信息增益的数字阈值用于进行分割。



虽然 C5.0 算法使用了信息增益，但信息增益并不是构建决策树的唯一分割标准。其他常用的标准有**基尼系数** (Gini index)、**卡方统计量** (Chi-Squared statistic) 和**增益比** (gain ratio)。对于这些（以及更多）标准的详细介绍，可以参阅 An Empirical Comparison of Selection Measures for Decision-Tree Induction, Machine Learning, Vol. 3, pp. 319-342, by J. Mingers (1989)。

2. 修剪决策树

一棵决策树可以继续无限制地增长，选择需要分割的特征，分割成越来越小的分区直到每一个案例完全归类，或者算法中再也没有可用于分割的特征。然而，如果决策树增长过大，将会使许多决策过于具体，模型将会过度拟合训练数据。而**修剪**一棵决策树的过程涉及减小它的大小，以使决策树能更好地推广到未知的数据。

解决这个问题的一种方法就是一旦决策树达到一定数量的决策，或者决策节点仅含有少量的案例，我们就停止树的生长，这叫做提前停止法，或者**预剪枝**决策树法。由于这种预剪枝决策树避免了做不必要的工作，所以这是一个有吸引力的策略。然而，不足之处在于没有办法知道决策树是否会错过细微但很重要的模式，这种细微的模式只有决策树生长到足够大时才能学习到。

另一种方法称为**后剪枝**决策树法，如果一棵决策树生长得过大，就根据节点处的错误率使用修剪准则将决策树减小到更合适的大小。该方法通常比预剪枝法更有效，因为如果没有事先生成决策树，要确定一棵决策树生长的最佳程度是相当困难的，而事后修剪决策树肯定可以使算法查到所有重要的数据结构。



修剪操作的实施细节是很具技术性的，超出了本书的范围。对于一些可用方法的比较，可以参阅：A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, pp. 476-491, by F. Esposito, D. Malerba, and G. Semeraro (1997)。

C5.0 算法的优点之一就是它可以自动修剪，即它关注许多决策，能自动使用相当合理的

默认值。该算法的总体策略就是事后修剪决策树，它先生成一个过度拟合训练数据的大决策树，然后删除对分类误差影响不大的节点和分枝。在某些情况下，整个分枝会被进一步向上移动或者被一些简单的决策所取代，这两个移植分枝的过程分别称为子树提升和子树替换。

权衡过度拟合与不足拟合一棵决策树是一门学问，但如果模型的准确性是至关重要的，那么就值得在不同的剪枝方案上花些时间，看看它是否对测试数据的性能有所改善，很快你就会看到，C5.0 算法的优点之一就是它很容易调整训练方案。

5.2 例子——使用 C5.0 决策树识别高风险银行贷款

2007–2008 年的全球金融危机凸显了透明度和严密性在银行业务中的重要性。由于信贷供应受到了限制，所以银行正日益紧缩其贷款体系，转向机器学习来更准确地识别高风险贷款。

因为决策树的准确性高，以通俗易懂的方法建立统计模型的能力强，所以它广泛地应用于银行业。由于许多国家的政府机构密切监控贷款业务，所以银行的高管必须能够解释为什么一个申请者被拒绝贷款申请，而其他入获得批准。此信息对于希望判断为何自己的信用评级是不符合要求的消费者也是有用的。

通过电话和网络，将自动化的信用评分模型用于即时进行的信贷审批是很有可能。在本节中，我们将使用 C5.0 决策树建立一个简单的信贷审批模型。我们也将看到应该如何调整模型的结果，从而使导致机构财务损失的误差最小化。

5.2.1 第 1 步——收集数据

我们信贷模型背后的思想就是确定使得申请者违约风险较高的因素，因此，我们需要获得大量的过去银行贷款的数据，以及贷款是否违约，同时还需要申请者的信息。

具有这些特征的数据可以从 UCI 机器学习数据仓库 (Machine Learning Data Repository) (<http://archive.ics.uci.edu/ml>) 的一个数据集得到，这些数据表示从德国的一个信贷机构获得的贷款，数据由汉堡大学的 Hans Hofmann 捐赠。



本章中给出的数据已经根据原始数据略微做了修正，因此可以省去一些预处理步骤。要理解这些例子，你需要从 Packt 出版社的网站下载 `credit.csv` 文件，并将该文件保存到 R 工作目录中。

该信贷数据集包含了 1000 个贷款的案例、一个用来表示贷款特征和贷款申请者特征的数值特征与名义特征的组合。一个类变量表示贷款是否陷入违约，让我们看看我们是否能够确定一些模型来预测贷款是否违约。

5.2.2 第 2 步——探索和准备数据

正如我们之前所做的，我们使用 `read.csv()` 函数导入数据。由于数据中的大多数特

征为名义特征（变量），所以我们将忽略 `stringsAsFactors` 选项（即使用默认值 `TRUE`）。我们还将研究我们所创建的信贷数据框的结构：

```
> credit <- read.csv("credit.csv")
> str(credit)
```

函数 `str()` 输出结果的前几行如下所示：

```
'data.frame':1000 obs. of 17 variables:
 $ checking_balance : Factor w/ 4 levels "< 0 DM", "> 200 DM", ...
 $ months_loan_duration: int 6 48 12 ...
 $ credit_history      : Factor w/ 5 levels "critical", "good", ...
 $ purpose            : Factor w/ 6 levels "business", "car", ...
 $ amount             : int 1169 5951 2096 ...
```

我们看到了预期的 1000 个观察值和 17 个特征（变量），这是因子和整数数据类型的组合

让我们来看看 `table()` 函数对贷款的一对特征输出的一些结果，似乎很有可能预测出违约者。特征（变量）`checking_balance` 和 `savings_balance` 表示支票和储蓄账户余额，并记录为分类变量：

```
> table(credit$checking_balance)
< 0 DM  > 200 DM 1 - 200 DM  unknown
    274      63      269      394
> table(credit$savings_balance)
< 100 DM > 1000 DM 100 - 500 DM 500 - 1000 DM  unknown
    603      48      103      63      183
```

由于贷款数据是从德国获取的，所以货币以德国马克（Deutsche Mark, DM）为单位。较大的支票和储蓄账户余额应该与较小的贷款违约可能性相联系，这看起来是一个安全的假设。

有些贷款的特征是数值型变量，比如贷款期限（`months_loan_duration`）和信贷申请的金额（`amount`）。

```
> summary(credit$months_loan_duration)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.0   12.0   18.0   20.9   24.0   72.0
> summary(credit$amount)
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 250   1366   2320   3271   3972   18420
```

贷款金额介于 250 ~ 18 420 马克之间，贷款期限为 4 ~ 72 个月，其贷款期限的中位数为 18 个月，贷款金额的中位数为 2320 马克。

变量 `default` 表示贷款申请者是否未能符合约定的付款条件而陷入违约。所有申请贷款的有 30% 陷入违约：

```
> table(credit$default)
no yes
700 300
```

银行贷款给违约率高的客户是不可取的，因为这意味着很有可能银行不能完全收回它的投资。如果我们成功，我们的模型将能识别可能违约的申请者，从而减少违约的数量。

数据准备——创建随机的训练数据集和测试数据集

正如我们在前面的章节中所做的那样，我们将数据分成两部分：用来建立决策树的训练数据集和用来评估模型性能的测试数据集。我们将使用 90% 的数据作为训练数据，10% 的数据作为测试数据，这将为我们提供 100 条记录来模拟新的申请者。

由于之前章节中使用的数据已经以随机的顺序排序，所以我们通过取前面 90% 的记录作为训练数据，剩下的 10% 作为测试数据，简单地将数据划分成了两部分。与之相反，我们这里的数据并不是随机排列的。假设银行已经根据贷款金额对数据进行了排序，最大金额的贷款排在文件的最后，如果我们使用前 90% 的数据作为训练数据，剩下的 10% 作为测试数据，那么我们将只根据小额贷款建立模型，而基于大额贷款测试模型。很明显，这是有问题的。

在划分数据前，我们通过随机排列信贷数据框来解决这个问题。`order()` 函数以升序或者降序的方式对数据列表进行重新排列，如果我们将 `order()` 函数与生成一系列随机数的函数相结合，就可以生成一个随机排序的列表。对于随机数的产生，我们可以使用 `runif()` 函数，该函数在默认情况下产生 0 ~ 1 之间的随机数序列。



如果你想了解函数 `runif()` 名字的由来，答案是因为该函数事实上是根据均匀分布产生的随机数，这在第 2 章中，我们已经学过。

下面的命令创建一个随机排序的 `credit` 数据框。`set.seed()` 函数可用来在一个预定义的序列中生成随机数，从一个称为**随机数种子**（这里设置为任意值 12345）的位置开始。这似乎违背了生成随机数的目的，但这样做有一个很好的理由，`set.seed()` 函数能够确保如果要重复这里的分析，那么可以获得相同的结果。

```
> set.seed(12345)
> credit_rand <- credit[order(runif(1000)), ]
```

命令 `runif(1000)` 生成了一个含有 1000 个随机数的列表。我们确实需要 1000 个随机数，因为在 `credit` 数据框中有 1000 条记录。然后，`order()` 函数返回一个数值向量，该向量记录了排序之后的 1000 个随机数在原来序列中的位置，我们可以利用这些位置来筛选 `credit` 数据框中的行，并保存在一个名为 `credit_rand` 的新数据框中。



为了更好地理解该函数的作用，注意 `order(c(0.25, 0.5, 0.75, 0.1))` 返回的序列为 4 1 2 3，因为最小的数 (0.1) 出现在第四个，第二小的数 (0.25) 出现在第一个，以此类推。

为了确认我们得到的是相同的数据，只是排序不同，我们在两个数据框之间比较特征

amount。下面的代码显示了主要的统计量：

```
> summary(credit$amount)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   250   1366   2320   3271   3972   18420

> summary(credit_rand$amount)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   250   1366   2320   3271   3972   18420
```

我们可以使用 `head()` 函数来查看每一个数据框中的前几个值：

```
> head(credit$amount)
[1] 1169 5951 2096 7882 4870 9055
> head(credit_rand$amount)
[1] 1199 2576 1103 4020 1501 1568
```

虽然前几个值不同，但由于主要的统计量是相同的，所以这表明我们随机排序的做法是正确的。



如果你的结果不能与之前的数据完全匹配，确保你在创建 `credit_rand` 数据框前及时运行命令 `set.seed(214805)`。

现在，我们可以将数据划分为训练数据（90% 或者 900 条记录）和测试数据（10% 或者 100 条记录），正如我们在之前的分析中所做的：

```
> credit_train <- credit_rand[1:900, ]
> credit_test  <- credit_rand[901:1000, ]
```

如果一切顺利，在每一个数据集中，我们应该有大约 30% 的违约贷款。

```
> prop.table(table(credit_train$default))
      no      yes
0.7022222 0.2977778

> prop.table(table(credit_test$default))
      no  yes
0.68 0.32
```

这似乎是一个相当平衡的分割，所以现在我们可以建立决策树了。

5.2.3 第 3 步——基于数据训练模型

我们将使用 C5.0 添加包中的 C5.0 算法来训练决策树模型。如果你还没有安装 C5.0 添加包，先要用命令 `install.packages("C5.0")` 来安装该添加包，用命令 `library(C5.0)` 将其加载到 R 的会话中。

下面的语法框列出了用于构建决策树的一些最常用的命令。与我们之前使用过的机器学习方法相比，C5.0 算法提供了更多的调整一个特定学习问题模型的方法，有更多的选项可供选择。命令 `?C5.0Control` 在帮助页面上显示了更多关于如何精细调整模型的细节。

对于我们信贷审批模型的第一次迭代，我们将使用默认的 C5.0 配置，如下面的代码所

示。在 `credit_train` 中，第 17 列是类变量 `default`，所以我们需要将它作为一个自变量从训练数据框中排除，但把它作为用于分类的目标因子向量。

```
> credit_model <- C5.0(credit_train[-17], credit_train$default)
```

C5.0 决策树语法

应用 C50 添加包中的函数 `C5.0()`

创建分类器：

```
m <- C5.0(train, class, trials = 1, costs = NULL)
```

- `train`: 一个包含训练数据的数据框
- `class`: 包含训练数据每一行的分类的一个因子向量
- `trial`: 为一个可选数值，用于控制自助法循环的次数（默认值为 1）
- `costs`: 为一个可选矩阵，用于给出与各种类型错误相对应的成本

该函数返回一个 C5.0 模型对象，该对象能够用于预测。

进行预测：

```
p <- predict(m, test, type = "class")
```

- `m`: 由函数 `C5.0()` 训练的一个模型
- `test`: 一个包含测试数据的数据框，该数据框和用来创建分类器的训练数据有同样的特征。
- `type`: 取值为 “class” 或者 “prob”，标识预测是最可能的类别值或者是原始的概率值。

该函数将返回一个向量，根据参数 `type` 的取值，该向量含有预测的类别值或者原始预测的概率值。

例子：

```
credit_model <- C5.0(credit_train, loan_default)
credit_prediction <- predict(credit_model, credit_test)
```

现在，对象 `credit_model` 包含一个 C5.0 决策树对象。我们可以通过输入其名称来查看关于该决策树的一些基本数据：

```
> credit_model

Call:
C5.0.default(x = credit_train[-17], y = credit_train$default)

Classification Tree
Number of samples: 900
Number of predictors: 16

Tree size: 67
```

前面的文字显示了一些关于该决策树的简单情况，包括生成决策树的函数调用、特征数（即 `predictions`）和用于决策树增长的案例（`Samples`）。同时列出树的大小为 67，这表明该树有 67 个决策，比我们迄今为止看到的决策树都要大很多。

要查看决策，我们可以对模型调用 `summary()` 函数：

```
> summary(credit_model)
```

结果输出如下：

```

c5.0 [Release 2.07 GPL Edition]
-----
Class specified by attribute 'outcome'
Read 900 cases (17 attributes) from undefined.data
Decision tree:
checking_balance = unknown: no (358/44)
checking_balance in {< 0 DM,> 200 DM,1 - 200 DM}:
:...credit_history in {perfect,very good}:
:...dependents > 1: yes (10/1)
:   dependents <= 1:
:   :...savings_balance = < 100 DM: yes (39/11)
:   :   savings_balance in {> 1000 DM,500 - 1000 DM,unknown}: no (8/1)
:   :   savings_balance = 100 - 500 DM:
:   :   :...checking_balance = < 0 DM: no (1)
:   :   :   checking_balance in {> 200 DM,1 - 200 DM}: yes (5/1)

```

上面的输出显示了决策树的前几个分枝，前 4 行可以用通俗易懂的语言来表达：

- 1) 如果支票账户余额是未知的，则归类为**不太可能违约**。
- 2) 否则，如果支票账户余额少于 0 马克，或者 1 ~ 200 马克之间，或者超过 200 马克，或者……
- 3) 信用记录非常好，甚至完美，或者……
- 4) 有多个受抚养人，就归类为**很有可能违约**。

括号中的数字表示符合该决策准则的案例的数量以及根据该决策不正确分类的案例的数量。例如，在第一行中，(358/44) 表示有 358 个案例符合该决策条件，有 44 个案例被错误地归类为 no，也就是说不太可能违约。换句话说，就是有 44 个申请者确实违约了，尽管模型的预测与此相反。



有些决策树中的决策似乎没有逻辑意义。为什么一个申请者的信用记录很好却很有可能违约，而那些支票余额未知的用户却不太可能违约呢？像这样矛盾的规则有时候会发生，它们可能反映了数据中的一个真实的模式，或者它们可能是统计中的异常值。

在决策树输出后，`summary(credit_model)` 输出一个混淆矩阵，这是一个交叉列表，表示模型对训练数据错误分类的记录数：

Evaluation on training data (900 cases):

```

Decision Tree
-----
Size      Errors
 66  125(13.9%)  <<

(a)  (b)  <-classified as
----  ----
 609    23  (a): class no
102   166  (b): class yes

```

字段 `Errors` 说明模型对除了 900 个训练案例中的 125 个案例以外的所有案例进行了正确的分类，错误率为 13.9%。共有 23 个真实值为 no 的个案被错误地归类为 yes(错误肯定)，

而有 102 个真实值为 yes 的个案被错误地归类为 no（错误否定）。

众所周知，决策树有一种过度拟合训练数据模型的倾向。由于这个原因，训练数据中报告的错误率可能过于乐观，因此，基于测试数据集来评估决策树模型是非常重要的。

5.2.4 第 4 步——评估模型的性能

为了将决策树应用于测试数据集，我们使用 `predict()` 函数，如下代码所示：

```
> credit_pred <- predict(credit_model, credit_test)
```

这样就创建了一个预测分类值向量，我们可以使用 `gmodels` 添加包中的 `CrossTable()` 函数将它与真实的分类值比较。设定参数 `prop.c` 和 `prop.r` 为 `FALSE` 来删除表中列与行的百分比，剩余的百分比（`prop.t`）表示单元格中的记录数占总记录数的百分比。

```
> library(gmodels)
> CrossTable(credit_test$default, credit_pred,
  prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
  dnn = c('actual default', 'predicted default'))
```

输出的结果如下表所示：

actual default	predicted default		Row Total
	no	yes	
no	57 0.570	11 0.110	68
yes	16 0.160	16 0.160	32
Column Total	73	27	100

在 100 个贷款申请者的记录测试中，我们的模型正确预测了 57 个申请者确实没有违约，而 16 个申请者确实违约了，模型的准确率为 73%，而错误率为 27%。在训练数据上模型的性能略差，但并不令人意外，因为对于未知的数据，模型的性能往往会差些。另外请注意，该模型只正确预测了测试数据中 32 个贷款违约者的 50%。遗憾的是，这种类型的错误是一个潜在的非常严重的错误。让我们看看，再多一点努力，我们是否能够改善预测结果。

5.2.5 第 5 步——提高模型的性能

我们模型的错误率很可能过高而不能将其应用于实时的信用评估申请中。事实上，如果该模型对每一个测试案例的预测结果为“没有违约”（no default），那么此时模型的正确率将为 68%——结果并不比我们的模型差太多，但需要少得多的努力！从 900 个案例中预测贷款违约似乎是一个具有挑战性的问题。

更糟糕的是，我们的模型在识别贷款违约申请者上的性能尤其不佳。幸运的是，有几种简单的方法可以用来调整 C5.0 算法，这可能有助于提高模型的性能，无论是整体性能还是对于更大代价的错误。

1. 提高决策树的准确性

C5.0 算法对 C4.5 算法改进的一种方法就是通过加入**自适应增强 (adaptive boosting)** 算法。这是许多决策树构建的一个过程，然后这些决策树通过投票表决的方法为每个案例选择最佳的分类。



boosting 算法的思想很大程度上建立在 Rob Schapire 和 Yoav Freund 的研究上。欲了解更多的信息，可尝试在网上搜索他们的文章或者他们最近的教学材：《Boosting: Foundations and Algorithms Understanding Rule Learners》(The MIT Press, 2012)。

由于 boosting 算法可以更广泛应用于任何机器学习算法，所以在本书的第 11 章中有该算法的详细信息。就目前而言，只需要说明 boosting 算法植根在这样一种概念中：通过将很多能力较弱的学习算法组合在一起，就可以创建一个团队，这比任何一个单独的学习算法都强得多。每一个模型都有一组特定的优点和缺点，对于特定的问题，可能更好，也可能更差，而使用优点和缺点互补的多种学习方法的组合，可以显著地提高分类器的准确性。

C5.0() 函数可以很轻松地将 boosting 算法添加到 C5.0 决策树中。我们只需要添加一个额外的参数 trials，表示在模型增强团队中使用的独立决策树的数量。参数 trials 设置了一个上限，如果该算法识别出额外的试验似乎并没有提高模型的准确性，那么它将停止添加决策树。我们将开始于 10 个试验 (trials=10)——一个已经成为事实标准的数字，因为研究表明，这能降低关于测试数据大约 25% 的错误率。

```
> credit_boost10 <- C5.0(credit_train[-17], credit_train$default,
                           trials = 10)
```

在查看由此生成的模型时，我们可以看到添加了一些用来标识变化的额外的行。

```
> credit_boost10
Number of boosting iterations: 10
Average tree size: 56
```

通过 10 次迭代，决策树变小。如果愿意，可以在命令提示符下，通过输入 summary(credit_boost10) 看到所有的 10 棵决策树。

让我们来看看我们训练数据的表现：

```
> summary(credit_boost10)

(a)      (b)      <-classified as
----      ----
626        6      (a): class no
25        243     (b): class yes
```

在 900 个训练案例中，该分类器犯了 31 个错误，错误率为 3.4%。我们注意到在加入 boosting 算法之前，训练案例的错误率为 13.9%。因此，与之前相比，这是很大的提高。然而，这仍有待观察，我们是否能在测试数据中看到类似的提高呢？让我们一起来看看：

```
> credit_boost_pred10 <- predict(credit_boost10, credit_test)
```

```
> CrossTable(credit_test$default, credit_boost_pred10,
  prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
  dnn = c('actual default', 'predicted default'))
```

输出的结果如下表所示：

actual default	predicted default		Row Total
	no	yes	
no	60 0.600	8 0.080	68
yes	15 0.150	17 0.170	32
Column Total	75	25	100

这里，在增强模型的性能后，我们将总的错误率由之前的 27% 下降到现在的 23%。这似乎并不像是一个大的增益，但正如我们所希望的，它合理地降低了近 25% 的错误率。另一方面，模型在预测贷款违约方面仍然做得不好，有 $15 / 32 = 47\%$ 是错误的。缺乏更大的提高可能是由于我们采用的是一个相对较小的训练数据集，也可能这本身就是一个很难解决的问题。

如上述所说，如果 boosting 算法可以很容易实现，那么为什么不在默认情况下，将它应用于每一棵决策树呢？原因有两方面。首先，如果建立一棵决策树需要花费大量的计算时间，那么建立很多决策树可能在计算上是不可行的。其次，如果训练数据很杂乱，那么 boosting 算法可能根本不会改进模型的性能。不过，如果需要更高的准确性，那还是值得尝试一下。

2. 犯一些比其他错误更严重的错误

给一个很有可能违约的申请者一笔贷款是一种很严重的错误。一种减少错误地否定申请者数量的方法是拒绝大量处于边界线的申请者。数年内，如果贷款者从来没有还钱，那么银行所蒙受的大量损失就远远超过了它从有风险的贷款者身上赚到的利息值。

为了防止一棵决策树犯更严重的错误，C5.0 算法能够允许我们将一个惩罚因子分配到不同类型的错误上。这些惩罚因子设定在一个代价矩阵中，用来指定每种错误相对于其他任何错误有多少倍的严重性。假设我们认为一个贷款违约者使银行损失了 4 倍，不亚于一个错失的机遇，则代价矩阵可以定义为：

```
> error_cost <- matrix(c(0, 1, 4, 0), nrow = 2)
```

这样就创建了一个 2 行 2 列的矩阵，它与我们一直研究的混淆矩阵相比，排列上稍有不同。行名和列名中的代码 1 表示 no，2 表示 yes。行用来表示预测值，列用来表示实际值：

```
> error_cost
      [,1] [,2]
[1,]    0    4
[2,]    1    0
```

正如该矩阵所定义的，当该算法正确地将一个案例分类为 no 或者 yes 时，此时没有分配代价，但是相对于代价为 1 的错误肯定 (false positive) 的案例，错误否定 (false negative) 的案例的代价为 4。为了看到这样做是如何影响分类的，我们将通过使用 C5.0() 函数中的参数

costs, 将其应用到我们的决策树中。我们将以其他方式应用与前面一样的步骤:

```
> credit_cost <- C5.0(credit_train[-17], credit_train$default,
                      costs = error_cost)
> credit_cost_pred <- predict(credit_cost, credit_test)
> CrossTable(credit_test$default, credit_cost_pred,
             prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
             dnn = c('actual default', 'predicted default'))
```

这将生成如下的混淆矩阵:

actual default	predicted default		Row Total
	no	yes	
no	42 0.420	26 0.260	68
yes	6 0.060	26 0.260	32
Column Total	48	52	100

与最好的增强模型相比, 这个版本的模型整体上犯了更多的错误: 相对于最好的增强模型的 23%, 这里为 32%。然而, 错误的类型却相差很大。前面的模型对几乎一半的贷款违约者错误分类, 而在这个模型中, 只有 25% 的贷款违约者被预测为非违约者。如果我们的代价估算是准确的, 那么以增加错误肯定为代价, 减少错误否定的这种折中是可以接受的。

5.3 理解分类规则

分类规则代表的是 if-else 逻辑语句形式的知识, 可用来对未标记的案例指定一个分类。未标记的案例依据前件和后件的概念而指定, 而前件和后件就构成了一个假设, 即“如果这种情况发生, 那么那种情况就会发生。”一个简单的规则或许会这样描述: “如果硬盘发出咔嗒声, 那么硬盘将出现故障。”前件是由特征值的特定组合构成的, 而在满足规则的条件下, 后件描述用来指定的分类值。

规则学习经常以一种类似于决策树学习的方式被使用。与决策树一样, 规则学习也可以用来为今后的行动形成认识, 比如:

- 确定导致机械设备出现硬件故障的条件。
- 描述人群的界定特征用于客户细分。
- 发现先于股票市场上股票价格大跌或者大涨的市况。

另一方面, 对于某些任务, 规则学习相对于决策树有明显的优势。与决策树不同的是, 决策树必须从上至下地应用, 而规则是单独存在的事实。根据相同数据建立的模型, 规则学习的结果往往比决策树的结果更简洁、更直接、更容易理解。



你将在本章的后面看到, 可以使用决策树生成规则。那么, 为什么我们还要费心单独研究规则学习算法呢? 原因是, 决策树会给任务带来一组特定的偏差, 而规则学习可以通过直接识别规则而避免偏差。

规则学习通常应用于以名义特征为主或全部是名义特征的问题。规则学习擅长识别偶发事件，即使偶发事件只是因为特征之间非常特殊的相互作用才发生的。

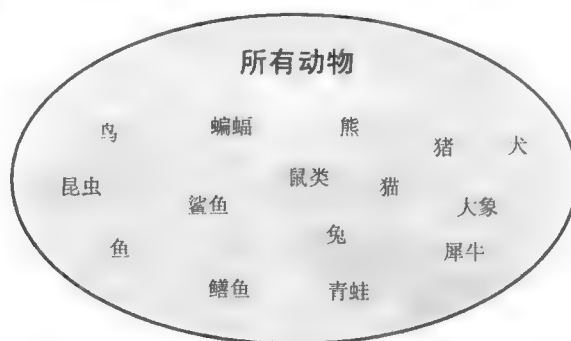
5.3.1 独立而治之

规则学习分类算法使用了一种称为**独立而治之**的探索法。这个过程包括确定训练数据中覆盖一个例子集的规则，然后再从剩余的数据中分离出该分区。随着规则的增加，更多的数据子集会被分离，直到整个数据集都被覆盖，不再有案例残留。

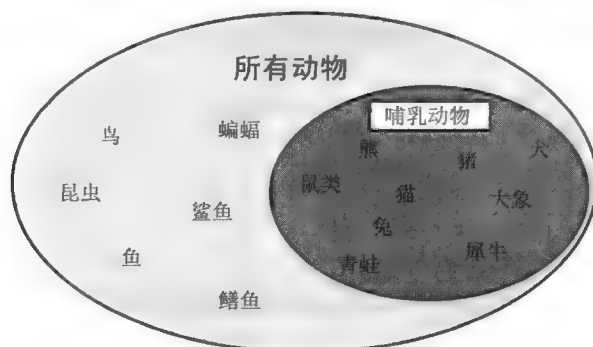


分而治之和独立而治之之间的区别是很小的。或许，区分这两种方法的最佳方式就是考虑决策树中每个决策节点是会受到过去决策历史的影响，而规则学习不存在这样的沿袭，一旦规则学习算法分离出一组案例，下一组案例可能会根据完全不同的特征，以完全不同的顺序分离出来。

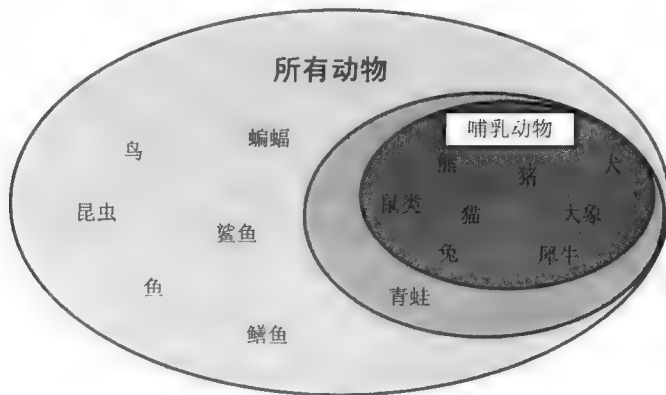
想象规则学习过程的一种方式就是通过创建用于标识分类值的越来越具体的规则来考虑向下挖掘（钻取）数据。假设任务是通过创建规则来确定一个动物是否是哺乳动物。你可以用一个大的空间来描述所有动物，如下图所示。



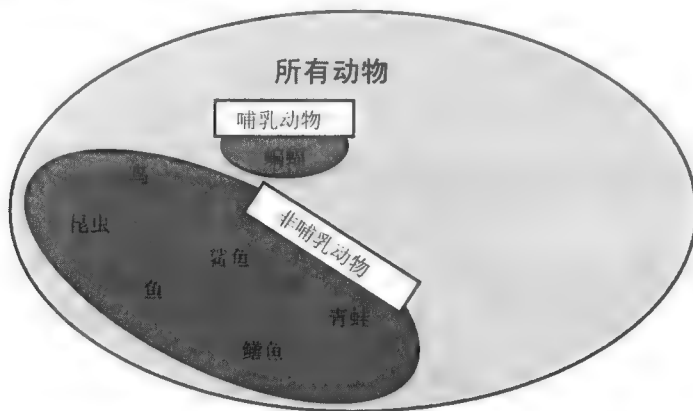
规则学习算法通过利用已有的特征来开始寻找同类群体。例如，根据衡量物种是在陆地、海洋，还是空中行走的特征，第一条规则可能表明任何陆地动物都是哺乳动物：



你是否注意到这条规则存在问题呢？如果仔细看，你可能会注意到青蛙是两栖动物，而不是哺乳动物。因此，规则需要更具体一点儿。让我们进一步细化规则，表明哺乳动物是在陆地上行走，并且有一条尾巴：



如上图所示，更具体的规则生成了一个全部是哺乳动物的子集。因此，该子集可以从其他数据中分离出来，而且可以通过定义额外的规则来确定剩余的哺乳动物蝙蝠。一个潜在的将蝙蝠从剩余的其他动物中区分出来的特征是其有皮毛。根据这个特征建立并利用这个规则，然后我们就能正确识别所有的动物：



这时，由于所有的训练案例都被分类了，所以规则学习过程就会停止。我们一共学习了3个规则：

- ☐ 在陆地上行走且有尾巴的动物是哺乳动物。
- ☐ 如果动物有皮毛，那么它就是哺乳动物。
- ☐ 否则，该动物不是哺乳动物。

前面的例子说明规则是如何逐步分离出越来越大的数据分区，最终将所有案例分类。因为数据的利用是基于先到先得的思想，所以分而治之和独立而治之算法又称为贪婪（学习）

算法。



贪婪算法一般都更有效率，但对于特定的数据集，它并不保证生成最佳的规则或者生成的规则数量最少。

由于规则看起来覆盖部分数据，所以独立而治之算法又称为**覆盖算法**，规则称为覆盖规则。下一节将通过研究一个简单的规则学习算法来学习覆盖规则在实际中是如何应用的。然后，我们将研究一个较复杂的规则学习算法，并将这两个算法都应用于现实世界的问题。

5.3.2 单规则 (1R) 算法

设想作为电视游戏节目的一部分，有一个均匀分成 10 块并涂色的轮盘，其中，3 块涂成红色，3 块涂成蓝色，还有 4 块涂成白色。在轮盘转动之前，你会被要求选择其中一种颜色。当轮盘停止转动时，如果显示的颜色与你的预测相符，你就会赢得一份大额度的现金奖励。你应该选择什么颜色呢？

如果选择白色，你当然更容易赢得奖励，因为这是轮盘上最常见的颜色。很显然，这个游戏节目有点可笑，但它演示了最简单的分类器 **ZeroR**，一个规则学习算法，从字面上看没有规则学习（因此而得名）。对于每一个未标记的案例，不用考虑它的特征值就会把它预测为最常见的类。

单规则算法（1R 或者 OneR）通过选择一个单一的规则来提高 ZeroR 算法的性能。虽然这看起来可能过于简单，但是它往往表现得比你预期的要好。正如 Robert C. Holte 在 1993 年的论文“Very Simple Classification Rules Perform Well on Most Commonly Used Datasets” (in Machine Learning, Vol. 11, pp. 63-91) 中所讲的，对于许多现实世界的任务，该算法的准确性可以接近于许多更复杂算法的准确性。该算法的优点和缺点如下表所示。

优点	缺点
<ul style="list-style-type: none"> 可以生成一个单一的、易于理解的、人类可读的经验法则（大拇指规则） 往往表现得出奇地好 可以作为更复杂算法的一个基准 	<ul style="list-style-type: none"> 只使用了一个单一的特征 可能会过于简单

该算法运作的方式很简单。对于每一个特征，基于相似的特征值 1R 对数据分组。然后，对于每一个数据分组，该算法的预测类为占多数的类。规则错误率的计算基于每一个特征，犯最少错误的规则选为唯一的规则。

对于本节研究的动物数据，下表显示了该算法是如何运行的：

动物	行走途径	是否有皮毛	哺乳动物
蝙蝠	空中	是	是
熊	陆地	是	是

(续)

动物	行走途径	是否有皮毛	哺乳动物
鸟	空中	否	否
猫	陆地	是	是
狗	陆地	是	是
鱗鱼	海洋	否	否
大象	陆地	否	是
鱼	海洋	否	否
青蛙	陆地	否	否
昆虫	空中	否	否
猪	陆地	否	是
兔	陆地	是	是
鼠	陆地	是	是
犀牛	陆地	否	是
鲨鱼	海洋	否	否

行走途径	预测值	真实值	
空中	否	是	×
空中	否	否	
空中	否	否	
陆地	是	是	
陆地	是	是	
陆地	是	是	
陆地	是	是	
陆地	是	否	×
陆地	是	是	
陆地	是	是	
陆地	是	是	
陆地	是	是	
海洋	否	否	
海洋	否	否	
海洋	否	否	

注：用行走途径为规则：错误率=2/15。

是否有皮毛	预测值	真实值	
否	否	否	
否	否	否	
否	否	是	×
否	否	否	
否	否	否	
否	否	否	
否	否	是	×
否	否	是	×
否	否	否	
是	是	是	
是	是	是	
是	是	是	
是	是	是	
是	是	是	
是	是	是	

注：用是否有皮毛为规则：错误率 = 3/15。

根据 **Travels By** (行走途径) 这一特征，我们将数据分为 3 组：**Air** (空中)、**Land** (陆地) 和 **Sea** (海洋)，在 **Air** (空中) 组和 **Sea** (海洋) 组的动物被预测为非哺乳动物，而 **Land** (陆地) 组的动物被预测为哺乳动物，这样就导致了 2 个错误：蝙蝠和青蛙。根据 **Has Fur** (皮毛特征) 可以将动物分为两组，有皮毛的动物被预测为哺乳动物，而没有皮毛的动物被预测为非哺乳动物，一共出现 3 个错误：猪、大象和犀牛。由于 **Travels By** (行走途径) 特征导致了更少的错误，所以 1R 算法将会基于 **Travels By** (行走途径) 返回下面的“一个规则”：

- 如果该动物在空中行走，那么它就不是哺乳动物。
- 如果该动物在陆地上行走，那么它就是哺乳动物。
- 如果该动物在海洋中行走，那么它就不是哺乳动物。

在发现了唯一的最重要的规则后，该算法就会止于此。

很明显，该算法对于某些任务来说可能过于简单了。你想要一个只考虑单一症状的医疗诊断系统吗？或者你想要一个只基于单一因素来停止或者加速车的自动驾驶系统吗？对于这些类型的任务，一个更复杂的规则学习算法可能会有用。我们将在下一节中学习这种算法。

5.3.3 RIPPER 算法

早期的规则学习算法受到两个问题的困扰。第一，它们是出了名的速度慢，使得它们对

于越来越多的大数据 (Big Data) 问题效率很低; 第二, 对于噪声数据, 它们往往不准确。

解决这些问题的第一步是由 Johannes Furnkranz 和 Gerhard Widmer 在 1994 年的一篇论文 “Incremental Reduced Error Pruning” (Proceedings of the 11th International Conference on Machine Learning, pp. 70-77) 中提出来的。该增量减少误差修剪算法 (IREP) 使用了生成复杂规则的预剪枝和后剪枝方法的组合, 并在将案例从全部数据集分离之前进行修剪。虽然这种策略有助于提高规则学习算法的性能, 但往往还是决策树表现得更好。

随着 1995 年 William W. Cohen 发表了一篇具有里程碑意义的论文 “Fast Effective Rule Induction” (Proceedings of the 12th International Conference on Machine Learning, pp. 115-123), 规则学习算法又向前迈进了一步。这篇论文介绍了重复增量修剪 (Repeated Incremental Pruning to Produce Error Reduction, RIPPER) 算法, 它对 IREP 算法进行改进后再生成规则, 它的性能与决策树相当, 甚至超过决策树。



规则学习分类算法的演变并没有到此为止, 新的规则学习算法正快速地提出。文献调查表明在许多其他算法中, 有 IRPE++ 算法、SLIPPER 算法、TRIPPER 算法等。

如下表所示, RIPPER 规则学习算法的优点和缺点通常可与决策树比较, 其主要优点就是可能生成一个稍微精简的模型。

优点	缺点
<ul style="list-style-type: none"> • 生成易于理解的、人类可读的规则 • 对大数据集和噪声数据集有效 • 通常比决策树产生的模型更简单 	<ul style="list-style-type: none"> • 可能会导致违反常理或者专家知识的规则 • 处理数值型数据不太理想 • 性能有可能不如更复杂的模型

RIPPER 算法是规则学习算法经过多次迭代进化而来的, 是用于规则学习的有效探索方法的一个组合。由于该算法的复杂性, 所以其技术实现细节的讨论超出了本书的范围, 但是, 它可以笼统地理解为一个三步过程:

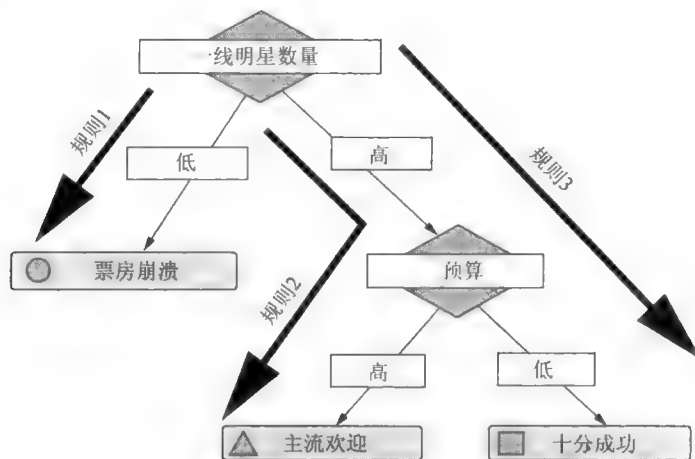
- 1) 生长。
- 2) 修剪。
- 3) 优化。

生长过程利用独立而治之技术, 对规则贪婪地添加条件, 直到该规则能完全划分出一个数据子集或者没有属性可用于分割。与决策树类似, 信息增益准则可用来确定下一个分割属性, 当增加一个特指的规则而熵值不再减小时, 该规则需要立即修剪。重复第一步和第二步, 直到达到一个停止准则, 然后, 使用各种探索法对整套的规则进行优化。

RIPPER 算法的多因素规则比 IR 算法的规则更复杂, 这意味着 RIPPER 算法可以考虑多个属性, 比如 “如果一个动物可以飞而且具有皮毛, 那么该动物就是哺乳动物。” 这样就提高了该算法对复杂数据建模的能力, 但与决策树一样, 这也意味着规则很快就会变得更加难以理解。

5.3.4 来自决策树的规则

分类规则也可以直接从决策树获得。从一个叶节点开始沿着树枝回到树根，将获得一系列的决策，这些决策可以组合成一个单一的规则。下图显示了如何根据决策树构建规则来预测成功的电影。



沿着从根节点到每个叶子的路径，规则将是：

- 1) 如果名人的数量少，那么该电影将属于 **Box Office Bust**（票房崩溃）类。
- 2) 如果名人的数量多且预算高，那么该电影将属于 **Mainstream Hit**（主流欢迎）类。
- 3) 如果名人的数量多且预算低，那么该电影将属于 **Critical Success**（十分成功）类。

使用决策树生成规则的主要缺点是由此产生的规则通常比那些由规则学习算法学到的规则更复杂。决策树应用分而治之策略产生的结果是有偏的，与规则学习产生的结果不同。另一方面，从决策树生成的规则有时候计算上会更有效。



当训练模型时，如果你指定 `rules=TRUE`，`C5.0()` 函数就可以利用分类规则生成一个模型。

5.4 例子——应用规则学习识别有毒的蘑菇

每年都会有很多人因为摄入有毒的野生蘑菇生病，有时甚至死亡。由于许多蘑菇在外观上彼此都非常相似，所以有时甚至经验丰富的蘑菇采集者都会中毒。

与识别其他有毒的植物（比如有毒的橡树或者有毒的常春藤）不一样，识别一种野生蘑菇是否有毒或者是否可以食用并没有明确的规则，如（有毒的常春藤）“三片叶子，不要碰它们（leaves of three, let them be）”。更加复杂的是，许多传统规则，比如“有毒的蘑菇颜色鲜艳”，提供的是危险的或者具有误导性的信息。如果有简单、清晰、一致的规则可用来识别

有毒的蘑菇，那么就可以拯救食物采集者的生命

由于规则学习算法的优势之一就是它们能生成易于理解的规则，所以规则学习算法似乎很适合这种分类任务。然而，规则只有在它们准确时才有用。

5.4.1 第1步——收集数据

为了确定用来区分有毒蘑菇的规则，我们将使用由卡内基梅隆大学的 Jeff Schlimmer 捐赠给 UCI 机器学习数据仓库 (Machine Learning Data Repository) 的蘑菇数据集 (Mushroom dataset)。原始数据可从网站 <http://archive.ics.uci.edu/ml/datasets/Mushroom> 获取

该数据集包括了列于 Audubon Society Field Guide to North American Mushrooms (1981) 上的 23 个带菌褶的蘑菇品种的 8124 个蘑菇案例信息。在食用指南中，每种蘑菇被鉴定为“肯定可以食用”、“肯定是有毒的”和“可能有毒，不建议食用”。对于该数据集的目的而言，最后一类和“肯定是有毒的”一类合并到一起，从而最终有两个类：有毒和无毒。UCI 网站提供的数据库字典描述了蘑菇案例的 22 个特征，包括的特征有蘑菇帽的形状、蘑菇帽的颜色、蘑菇的气味、菌褶的大小和颜色、茎的形状和生存的环境等。



本章使用的蘑菇数据是稍微修正过的版本。如果你打算一起学习这个例子，那么你需要从 Packt 出版社的网站下载 mushrooms.csv 文件，并将该文件保存到 R 的工作目录下。

5.4.2 第2步——探索和准备数据

首先，我们使用 `read.csv()` 函数导入数据来进行分析。由于所有 22 个特征和目标类都是名义变量，所以在这种情况下，设置 `stringsAsFactors=TRUE`，并采用自动因子转换：

```
> mushrooms <- read.csv("mushrooms.csv", stringsAsFactors = TRUE)
```

正如数据库字典所描述的，`str(mushrooms)` 命令的输出显示了该数据集包含的 23 个变量的 8124 个观测值的结构信息。虽然函数 `str()` 的大多数输出是很平常的，但有一个特征值得一提，注意下一行中关于 `veil_type` 变量有什么特别之处吗？

```
$ veil_type : Factor w/ 1 level "partial": 1 1 1 1 1 1 ...
```

如果你认为一个因素变量只有一个水平值是很奇怪的，那么你就对了。数据库字典列出了这个特征的两种水平：`partial` 和 `universal`，然而我们数据中的所有案例都分类为 `partial`。很可能，这个变量的有些编码是不正确的。由于变量 `veil_type` 的取值对所有案例都是一样的，所以它不能为预测提供任何有用的信息。使用下面的命令，将该变量从我们的分析中删除：

```
> mushrooms$veil_type <- NULL
```

通过将 `NULL` 赋给变量 `veil_type`，R 从蘑菇数据框中删除了这个变量。

在进一步研究之前，我们需要快速查看数据集中蘑菇类型这一类变量的分布。如果分类水平的分布很不均匀（这意味着它们严重失衡），则有些模型，比如规则学习，在预测少数类时会有困难。

```
> table(mushrooms$type)
edible poisonous
4208      3916
```

大约 52% 的蘑菇案例（ $N=4208$ ）是可食用的，而大约 48% 的蘑菇案例（ $N=3916$ ）是有毒的。由于分类水平大致被分割成 50/50，所以我们并不需要担心数据的不平衡。

为了达到这次试验的目的，我们把蘑菇数据中的 8124 个案例看做一个所有可能的野生蘑菇的完备集。这是一个重要的假设，因为该假设意味着我们不需要从训练数据中保存一些案例来达到测试的目的。我们没有尝试研究规则来覆盖不可预测的蘑菇类型，我们只是试图找到能准确描绘已知蘑菇类型这一完备集的规则。因此，我们可以依据相同的数据来建立模型并测试模型。

5.4.3 第 3 步——基于数据训练模型

如果我们基于该数据训练一个假想的 ZeroR 分类器，那么该分类器会做出什么样的预测呢？由于 ZeroR 忽略了所有的特征，只是预测目标的模式，所以用通俗易懂的话说，就是它的规则会这样陈述：“所有的蘑菇都是可食用的。”显然，这不是一个很有用的分类器，因为它会导致蘑菇采集者生病或者死亡于近一半的蘑菇案例。而我们的规则需要比该基准好得多，所以可以提供能够发布的安全建议。同时，我们也需要很容易记住的简单规则。

由于简单规则通常极具预测性，所以让我们看看对于 mushroom 数据，一个非常简单的规则学习算法是如何表现的。为此，我们将应用 1R 分类器，它能够识别对于目标类最具有预测性的单一特征，并利用该特征构建一个规则集。

我们将使用 RWeka 包中称为 OneR() 的函数来实现 1R 算法。你可能还记得在第 1 章中，我们已经安装了 RWeka 包。如果你还没有安装它，你需要使用 `install.packages("RWeka")` 命令，并在你的系统上安装 Java（请参考安装说明了解更多详细信息）。这些步骤完成后，通过键入 `library(RWeka)` 来加载 RWeka 包。

OneR() 使用 R 中的公式语法来指定要训练的模型。其公式语法使用运算符 `~`（称为波浪号）表示一个目标变量和它的预测变量之间的关系。需要学习的类变量放在波浪号的左侧，预测变量写在波浪号的右侧，用运算符 `+` 分隔。如果你想在类变量 y 和预测变量 x_1 与 x_2 之间建立关系，你需要写成公式： $y \sim x_1 + x_2$ 。如果你想在模型中包含所有的变量，你可以使用专业术语 `~.`。例如， $y \sim .$ 指定 y 和数据集中所有其他特征的关系。



R 中的公式语法被 R 中的很多函数使用，并提供了一些强大的功能来描述预测变量之间的关系。我们将会在后面的章节中探讨其中的一些功能，然而，如果你渴望先睹为快，可随时使用 `?formula` 命令来阅读帮助文件。

1R 分类规则语法

应用 RWeka 添加包中的函数 OneR()

创建分类器:

```
m <- OneR(class ~ predictors, data = mydata )
```

- `class`: 是 `mydata` 数据框中需要预测的那一列
- `predictors`: 为一个公式, 用来指定 `mydata` 数据框中用来进行预测的特征
- `data`: 为包含 `class` 和 `predictors` 所要求的数据的数据框

该函数返回一个 1R 模型对象, 该对象能够用于预测

进行预测:

```
p <- predict(m, test)
```

- `m`: 由函数 `OneR()` 训练的一个模型
- `test`: 一个包含测试数据的数据框, 该数据框和用来创建分类器的训练数据有同样的特征。

该函数将返回一个含有预测的类别值的向量。

例子:

```
mushroom_classifier <- OneR(type ~ odor + cap_color,
                             data = mushroom_train)
mushroom_prediction <- predict(mushroom_classifier,
                                mushroom_test)
```

使用公式 `type~.`, 当构建算法规则来预测 `type` 时, 将允许第一个规则学习算法 `OneR()` 考虑 `mushroom` 数据中所有可能的特征:

```
> mushroom_1R <- OneR(type ~ ., data = mushrooms)
```

为了查看该算法创建的规则, 可以输入分类器对象的名称, 在这种情况下, 输入 `mushroom_1R`:

```
> mushroom_1R
```

odor:

```
almond  -> edible
anise   -> edible
creosote -> poisonous
fishy   -> poisonous
foul    -> poisonous
musty   -> poisonous
none    -> edible
pungent -> poisonous
spicy   -> poisonous
```

```
(8004/8124 instances correct)
```

在输出的第一行, 我们看到特征 `odor` (气味) 被选为规则生成。特征 `odor` 的类别, 比如 `almond` (杏仁味)、`anise` (茴香味) 等, 注明蘑菇是否是可食用的或者是有毒的规则。例如, 如果蘑菇闻起来 `fishy` (腥味)、`foul` (臭味)、`musty` (霉味)、`pungent` (刺鼻)、`spicy` (辛辣) 或者像 `creosote` (木焦油), 那么该蘑菇很可能是有毒的。另一方面, 更加令人愉悦的气味, 像 `almond` (杏仁味)、`anise` (茴香味) (或者 `none`, 根本没有气味), 则表明是可食用的蘑菇。对于蘑菇采集的食用指南, 这些规则可能归纳在一个简单的经验

规则（大拇指规则）里：“如果蘑菇闻起来不好吃，那么它很可能是有毒的。”

5.4.4 第4步——评估模型的性能

输出的最后一行表明该规则正确地指定了 8124 个蘑菇案例中的 8004 个案例，正确率接近 99%。我们可以使用 `summary()` 函数获得关于分类器的更多详细信息，如下面的例子所示：

```
> summary(mushroom_1R)

=== Summary ===
Correctly Classified Instances      8004   98.5229 %
Incorrectly Classified Instances    120    1.4771 %
Kappa statistic                     0.9704
Mean absolute error                 0.0148
Root mean squared error             0.1215
Relative absolute error             2.958 %
Root relative squared error         24.323 %
Coverage of cases (0.95 level)     98.5229 %
Mean rel. region size (0.95 level)  50 %
Total Number of Instances          8124

=== Confusion Matrix ===
   a    b   <-- classified as
4208    0 |    a = edible
 120 3796 |    b = poisonous
```

标记为 `Summary` 的这部分列出了用来衡量 1R 分类器性能的很多不同的指标。由于我们将在第 10 章中介绍其中很多的统计量，所以现在我们将忽略它们。

标记为 `Confusion Matrix` 的这部分的类似于之前使用的混淆矩阵。这里，我们可以看到规则在哪里出现了问题，表中列表示蘑菇的真实类别，而表中的行表示预测值。注释显示在右侧，用 `a=edible` 和 `b=poisonous` 表示。左下角的值 120 表示有 120 种蘑菇实际上是可以食用的，但被归类为是有毒的。另一方面，没有有毒的蘑菇被错误地归类为可食用的。

基于该信息，似乎我们的 1R 规则实际上起着安全的作用——当觅食蘑菇时，如果你避免倒胃口的气味，那么你将避免吃到任何有毒的蘑菇。然而，你可能会错过一些实际上可以食用的蘑菇。考虑到这个规则学习算法只使用了一个单一的特征，我们确实做得不错。关于蘑菇的下一个食用指南的出版商应该会很开心。不过，让我们看一看是否可以添加一些规则，从而开发出更好的分类器。

5.4.5 第5步——提高模型的性能

对于一个更复杂的规则学习算法，我们将使用 `JRip()` 函数，一个基于 Java 实现的 RIPPER 规则学习算法。与我们之前使用的 1R 算法的实现一样，`RWeka` 包中已包含了 `JRip()` 函数。如果你还没有加载 `RWeka` 添加包，请务必使用 `library(RWeka)` 命令来加载该添加包。

RIPPER 分类规则语法

应用 RWeka 添加包中的函数 JRip()

创建分类器:

```
m <- JRip(class ~ predictors, data = mydata )
```

- **class**: 是 mydata 数据框中需要预测的那一列
- **predictors**: 为一个 R 公式, 用来指定 mydata 数据框中用来进行预测的特征
- **data**: 为包含 class 和 predictors 所要求的数据的数据框

该函数返回一个 RIPPER 模型对象, 该对象能够用于预测

进行预测:

```
p <- predict(m, test)
```

- **m**: 由函数 JRip() 训练的一个模型
 - **test**: 一个包含测试数据的数据框, 该数据框和用来创建分类器的训练数据有同样的特征
- 该函数将返回一个含有预测的类别值的向量。

例子:

```
mushroom_classifier <- JRip(type ~ odor + cap_color,
                             data = mushroom_train)
mushroom_prediction <- predict(mushroom_classifier,
                                mushroom_test)
```

如语法框中所示, 训练 JRip() 模型的过程与我们之前训练 OneR() 模型的过程是类似的。这是 RWeka 包中函数非常好的优点之一, 而且算法之间的语法是一致的, 这样使得比较许多不同模型的过程变得很简单。

让我们来训练 JRip() 规则学习算法, 正如我们对 OneR() 算法所做的那样, 允许它从所有的可用特征中选择规则:

```
> mushroom_JRip <- JRip(type ~ ., data = mushrooms)
```

若要查看规则, 输入分类器的名称:

```
> mushroom_JRip
```

JRIP rules:

=====

```
(odor = foul) => type=poisonous (2160.0/0.0)
(gill_size = narrow) and (gill_color = buff) => type=poisonous (1152.0/0.0)
(gill_size = narrow) and (odor = pungent) => type=poisonous (256.0/0.0)
(odor = creosote) => type=poisonous (192.0/0.0)
(spore_print_color = green) => type=poisonous (72.0/0.0)
(stalk_surface_below_ring = scaly) and (stalk_surface_above_ring = silky)
=> type=poisonous (68.0/0.0)
(habitat = leaves) and (cap_color = white) => type=poisonous (8.0/0.0)
(stalk_color_above_ring = yellow) => type=poisonous (8.0/0.0)
=> type=edible (4208.0/0.0)
```

Number of Rules : 9

JRip() 分类器从 mushroom 数据中学习了 9 条规则。理解这些规则的一种简单方法就是把它们当做类似于编程逻辑中 if-else 语句的一个列表。前 3 条规则可以这样表达:

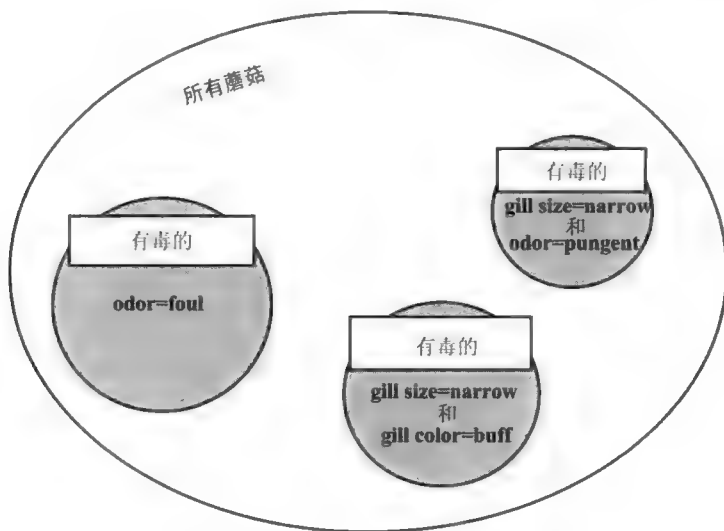
- 如果气味是臭的，那么该蘑菇类是有毒的。
- 如果菌褶的尺寸狭小而且菌褶的颜色是浅黄色的，那么该蘑菇类是有毒的。
- 如果菌褶的尺寸狭小而且气味是刺鼻的，那么该蘑菇类是有毒的。

最后，第9条规则表示不属于上述8条规则的任何蘑菇案例都是可食用的。根据编程逻辑的例子，这可以理解为：

- 否则，蘑菇是可食用的。

每个规则后面的数字表示被规则覆盖的案例数和被错误分类的案例数。值得注意的是，使用这9个规则就没有被错误分类的蘑菇案例。因此，被最后一个规则覆盖的案例数正好等于数据中可食用的蘑菇数量 ($N=4208$)。

下图给出了规则是如何应用于蘑菇数据的一个大致的说明。如果你把椭圆内的一切想象成所有的蘑菇品种，规则学习算法确定特征或者特征集，这样就将同类的分到了较大的组中。首先，该算法发现了一大群由它们的恶臭气味唯一区分开来的有毒的蘑菇；其次，该算法发现了较小的并且更具体的有毒的蘑菇群。通过确定规则覆盖每一种有毒的蘑菇品种，剩下的蘑菇就都是可食用的。感谢大自然母亲，由于每一种蘑菇品种都是足够独特的，所以分类器才能够达到100%的正确率。



5.5 总结

本章介绍了两种分类算法，它们根据特征的取值将数据进行分类。决策树使用分而治之策略来创建流程图，而规则学习使用独立而治之的数据来确定合乎逻辑的 if-else 规则。即使没有统计背景，这两种方法产生的模型都可以被理解。

一种流行的且可以灵活配置的决策树算法是 C5.0 算法。使用 C5.0 算法创建一棵决策树来预测一个贷款申请者是否会违约。使用 `boosting` 选项和 `cost-sensitive`(成本敏感) 误差选项, 能够提高模型的准确性, 并且可以避免可能消耗银行更多钱的高风险贷款。

我们还使用了两个规则学习算法, 1R 算法和 RIPPER 算法, 来研究用于识别有毒蘑菇的规则。1R 算法使用了一个单一的特征, 在确认很有可能致命的蘑菇案例时, 达到了 99% 的正确率。另一方面, 由更复杂的 RIPPER 算法生成的一组 9 个规则能够正确识别每一种蘑菇的可食用性。

本章仅仅涉及如何应用决策树和规则的表面。第 6 章介绍称为回归树和模型树的方法, 使用决策树预测数值型数据。第 11 章将发现, 决策树的性能可以通过将它们组合在一个称为随机森林的模型中而得到提高。第 8 章将看到关联规则——分类规则的关联, 如何被用来识别交易数据中的商品组。



预测数值型数据——回归方法

数学关系可以用来描述日常生活的许多方面。例如，一个人的体重可以从他的卡路里摄入量来描述；一个人的收入与他的受教育年限和工作经验相关；再次当选总统的胜率可以通过民意投票的支持率估计。

在每个这样的例子中，数字都精确地说明了数据元素是如何相关的。每天食用额外的 25 万卡路里很可能导致每个月增加将近 1 公斤的体重；每一年的工作经验在年薪中可能会值额外的 1000 美元，而受教育年限可能会值 2500 美元；如果一个总统具有很高的支持率，那么他更加有可能连任。显然，这些类型的方程并不能完美地模拟每个案例，但一般而言，这些规则可能会起到相当好的效果。

在统计领域中，有很大一部分工作介绍用于估计数据元素之间这种数值关系的方法，其中一个研究领域称为回归分析。这些方法可用于预测数值型数据以及量化预测结果与其预测变量之间关系的大小及强度。

本章将学习如何将回归方法应用到你自己的数据中。沿着本章的思路，你将学习：

- 用线性回归方法来拟合数据方程的基本统计原则和它们如何描述数据元素之间的关系。
- 如何使用 R 准备数据进行回归分析，定义一个线性方程并估计回归模型。
- 如何使用称为回归树和模型树的混合模型，它使得决策树可用来预测数值型数据。

到现在为止，我们只研究了适用于分类的机器学习方法。本章中的方法可以让你解决一系列全新的学习任务。记住这一点，那我们开始吧。

6.1 理解回归

回归主要关注确定一个唯一的**因变量** (dependent variable) (需要预测的值) 和一个或多个

数值型的**自变量**（independent variables）（预测变量）之间的关系。我们首先假设因变量和自变量之间的关系遵循一条直线，即线性关系。



用来描述数据拟合线过程的“回归”（regression）一词来源于19世纪后期 Francis Galton 爵士遗传学的研究中。Galton 发现，尽管父亲的身高极矮或者极高，但是他们儿子的身高却有更接近于平均身高的趋势，于是，他称这种现象为“回归平均值”（regression to the mean）。

你可能还记得代数中是以类似于 $y=a+bx$ 的斜截式来定义直线的，其中， y 是因变量， x 是自变量。在这个公式中，**斜率**（slope） b 表示每增加一个单位的 x ，直线会上升的高度；变量 a 表示当 $x=0$ 时 y 的值，它称为截距，因为它指定了直线穿过垂直轴时的位置。

回归方程使用类似于斜截式的形式对数据建立模型。该机器的的工作就是确定 a 和 b ，从而使指定的直线最适合用来反映所提供的 x 值和 y 值之间关系，这可能不是完美的匹配，所以该机器也需要有一些方法来量化误差范围，我们很快就会深入讨论这个问题。

回归分析通常用来对数据元素之间的复杂关系建立模型，用来估计一种处理方法对结果的影响和推断未来。一些具体的应用案例包括：

- 根据种群和个体测得的特征，研究他们之间如何不同（差异性），从而用于不同领域的科学研究，如经济学、社会学、心理学、物理学和生态学。
- 量化事件及其相应的因果关系，比如可应用于药物临床试验、工程安全检测、销售研究等。
- 给定已知的准则，确定可用来预测未来行为的模型，比如用来预测保险赔偿、自然灾害的损失、选举的结果和犯罪率等。

回归方法也可用于假设检验，其中包括数据是否能够表明原假设更可能是真还是假。回归模型对关系强度和一致性的估计提供了信息用于评估结果是否是由于偶然性造成的。



由于假设检验在技术上并不是一种学习任务，所以我们不会很深入地介绍它。如果你对这个主题感兴趣，你可以从入门的统计学教科书开始学习。

与我们到目前为止已经介绍过的其他机器学习算法不同，回归分析并不等同于一个单一的算法。相反，它是大量方法的一个综合体，几乎可以应用于所有的机器学习任务。如果你被限制只能选择一种单一的分析方法，那么回归方法将是一个不错的选择。你可以投身整个事业生涯来专门研究这种方法，而不去管其他方法，即使如此你还有可能学不完。

在本章中，我们只关注最基本的回归模型，即那些使用直线回归的模型，这叫做**线性回归**（linear regression）。如果只有一个单一的自变量，那就是所谓的**简单线性回归**（simple linear regression），否则，称为**多元回归**（multiple regression），这两个模型都假设因变量是连续的。

对于其他类型的因变量，即使是分类任务，使用回归方法都是可能的。例如，**逻辑回归**（logistic regression）可以用来对二元分类的结果建模；而**泊松回归**（Poisson regression），以法国数学家 Siméon Poisson 的名字命名，可以用来对整型的计数数据建模。相同的基本原则适用

于所有的回归方法，所以一旦你理解了线性情况下的回归方法，你就可以研究其他的回归方法



线性回归、逻辑回归、泊松回归以及许多其他的回归都属于一类模型，称为**广义线性模型**（Generalized Linear Model, GLM），使得回归能适用于许多类型的数据。线性模型可以通过使用连接函数（link function）进行泛化，其中，连接函数指定 x 和 y 之间的数学关系。

尽管简单线性回归中有“简单”两个字，但并没有简单到不能解决复杂的问题。在下一节中，我们将看到应用简单线性回归模型如何可能避免一场本来可以避免的悲剧性的工程灾难。

6.1.1 简单线性回归

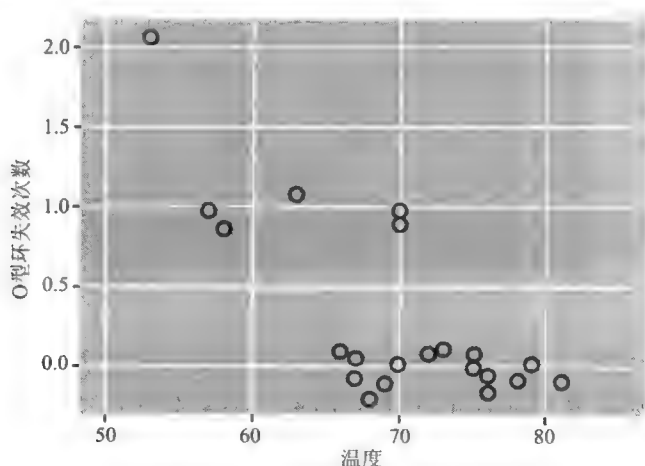
1986年1月28日，当负责密封火箭助推器关节的O型环失灵时，导致了一场灾难性的爆炸，美国挑战者号航天飞机的7名机组人员丧生。

那日夜晚之前，曾有过一场关于低温预报可能如何影响发射安全的长时间的讨论。航天飞机的部件从来没有在这样寒冷的天气测试过，因此，还不清楚设备是否能够经受由于严寒温度产生的形变。火箭工程师认为，寒冷的温度可能会使部件变得更脆，不太能恰当地密封，这将会导致危险燃料泄露的可能性较高。然而，考虑到政治压力要继续发射，他们需要数据来支持他们的假设。



本节的分析基于“Risk analysis of the space shuttle: pre-Challenger prediction of failure”，Journal of the American Statistical Association, Vol. 84, pp. 945-957, by S.R. Dalal, E.B. Fowlkes, and B. Hoadley, (1989) 提供的数据。

科学家的讨论转向之前23次航天飞机成功发射的数据，这些数据记录了O型环相对于发射温度失灵的次数。由于航天飞机共有6个O型环，所以每个O型环额外的失灵都会增加一场灾难性燃料泄漏的可能性。下面的散点图显示了这组数据。



研究该图，温度和失灵的数量之间存在着明显的趋势。在较高温度下发生的发射，O 型环往往几乎不会失效。此外，最低温度下（62 华氏度）的发射有两个环失效，是所有发射中失效个数最多的。因此，安排挑战者号在比 30 度还低的温度下发射似乎令人关注。为了从数量上说明该风险，我们可以应用简单线性回归。

简单线性回归定义了一个因变量和一个单一的自变量之间的关系，它由如下方程表示的一条直线来定义：

$$y=a+\beta x$$

不要被希腊字符吓坏，这个方程依然可以用之前所描述的斜截式来理解。截距 a （alpha）描述直线穿过垂直轴的位置，而斜率 β （beta）描述给定 x 的一个单位增加量后， y 的变化。对于航天飞机的发射数据，斜率将告诉我们发射温度每升高一度，O 型环失效数目的预期减少值。



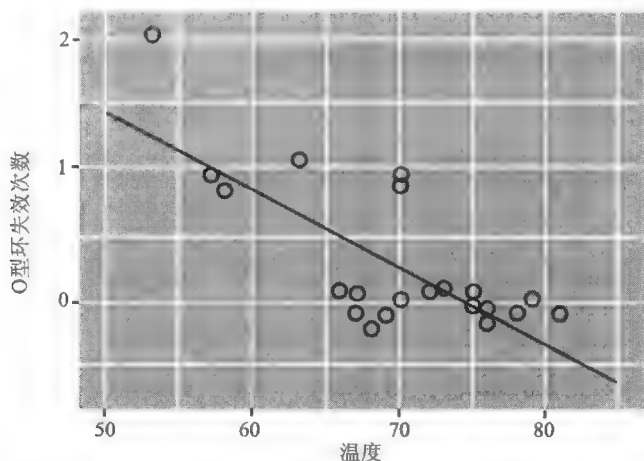
希腊字符通常用在统计领域，以表示一个统计函数的参数变量。因此，进行回归分析时，涉及对 a 和 β 寻找**参数估计**。 a 和 β 的参数估计值一般用 a 和 b 来表示，不过你可能发现这方面的一些数据和符号可以互换使用。

假设我们知道，对于航天飞机发射数据中回归方程的系数估计为：

$$\square a = 4.30$$

$$\square b = -0.057$$

因此，完整的线性方程为： $y = 4.30 - 0.057x$ 。先忽略这些数字是如何得到的，我们可以在散点图中画出这条直线：



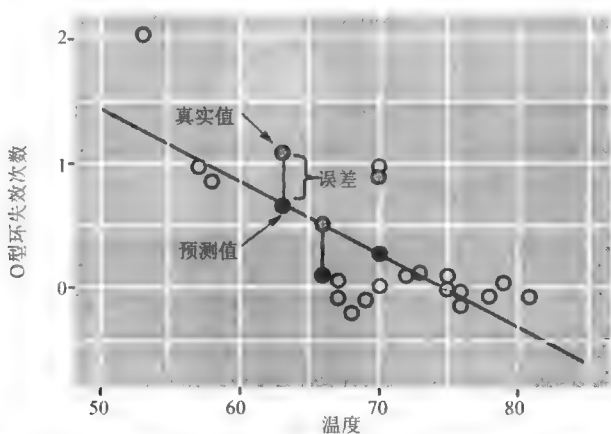
如直线所示，在华氏 60 度，我们预测 O 型环的失效数略低于 1，在华氏 70 度，我们预计失效数在 0.3 左右。如果应用我们的模型在华氏 31 度进行预测——即挑战者号发射的预测温度，那么我们预计有 $4.30 - 0.057 \times 31 = 2.53$ 个 O 型环失效。假设每个 O 型环失效导致一场

灾难性的燃料泄漏是等可能的，这意味着挑战者号的发射大约比在正常的华氏 60 度下发射的风险高出 3 倍多，而比在华氏 70 度下发射一次的风险高 8 倍多。

注意，该直线并不能准确地预测数据。相反，它略微均衡地穿过了数据，有些预测值比预期的要低，有些比预期的要高。下一节将学习为什么会选择这条特定的直线。

6.1.2 普通最小二乘估计

为了确定 a 和 b 的最优估计值，可使用一种称为普通最小二乘 (Ordinary Least Squares, OLS) 的估计方法。在 OLS 回归中，斜率和截距的选择要使得误差 (即 y 的预测值与 y 的真实值之间的垂直距离) 的平方和最小。这些误差称为残差 (residual)，可通过前一个图中的几个点来说明：



用数学中的术语，OLS 回归的目标可以表述为求下述方程最小值的任务：

$$\sum (y_i - \hat{y}_i)^2 = \sum e_i^2$$

用通俗易懂的语言来说，这个方程定义了 e (误差) 为真实值 y 和预测的 y 值之间的差值，需要将数据中的所有点的误差值平方并求和。



y 项上的插入符号 (^) 是统计表示法中一个常用的符号，它表示这一项是对真实值 y 的一个估计，称为 y -hat。

虽然证明超出了本书的范围，但是可以通过演算来证明使得平方误差最小的 b 值为：

$$b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

而 a 的最优值为：

$$a = \bar{y} - b\bar{x}$$



要理解这些方程，还需要知道另一个统计符号。出现在 x 项和 y 项上方的水平线表示 x 和 y 的均值，称为 \bar{x} 和 \bar{y} 。

为了理解这些方程，我们可以将它们分解来看。 b 值的分母应该看起来很熟悉，它类似于 x 的方差，而方差可以表示为 $\text{Var}(x)$ 。正如我们在第 2 章中所学到的，计算方差涉及求 x 与其均值的平均平方偏差。

我们之前还没有计算分子，分子涉及求每个数据点中 x 与其均值的偏差乘以 y 与其均值的偏差的乘积之和，这称为 x 和 y 的**协方差** (covariance)，表示为 $\text{Cov}(x, y)$ 。考虑到这一点，可以将 b 值的公式重新写成：

$$b = \frac{\text{Cov}(x, y)}{\text{Var}(x)}$$



如果你想一起学习这些例子，那么你需要从 Packt 网站下载 `challenger.csv` 文件，并使用命令 `launch <- read.csv("challenger.csv")` 将该文件加载并存储为一个数据框。

给定此公式，使用 R 中的函数很容易计算 b 的值。假设航天飞机发射的数据存储在一个名为 `launch` 的数据框中，自变量 x 为 `temperature`，因变量 y 为 `distress_ct`。然后，我们便可以使用 R 中的内置函数 `cov()` 和 `var()` 来估计 b ：

```
> b <- cov(launch$temperature, launch$distress_ct) /
      var(launch$temperature)
> b
[1] -0.05746032
```

从这开始，我们可以使用 `mean()` 函数估计 a ：

```
> a <- mean(launch$distress_ct) - b * mean(launch$temperature)
> a
[1] 4.301587
```

以这种方式估计回归方程是不理想的，所以 R 中理所当然地提供了可以自动估计回归方程的函数。我们很快就会研究这些函数。首先，我们将通过学习一种用来衡量线性关系强度的方法来拓展我们对回归的理解；然后，我们将看一看线性回归如何应用于有多个自变量的数据的中。

6.1.3 相关系数

两个变量之间的**相关系数** (correlation) 是一个数，它表示两个变量服从一条直线的关系有多么紧密。如果没有其他的限制，相关系数就是指 Pearson 相关系数，它是由 20 世纪数学家 Karl Pearson 提出来的，相关系数的范围是在 $-1 \sim +1$ 之间，两端的值表示一个完美的线

性关系，而相关系数接近于零则表示不存在线性关系。

下面的公式定义了 Pearson 相关系数：

$$\rho_{x,y} = \text{Corr}(x,y) = \frac{\text{Cov}(x,y)}{\sigma_x \sigma_y}$$



这里介绍一些希腊字符：第一个字符 ρ （看起来像小写字母“p”）是 rho，用来表示 Pearson 相关系数；第二个是分母上的字符 σ ，它看起来像是向一侧转过的“q”，记为 sigma， σ_x 、 σ_y 分别表示 x 、 y 的标准差。

使用这个公式，我们可以计算发射温度和 O 型环失效数之间的相关系数。回想一下，协方差函数为 `cov()`，标准差函数为 `sd()`。我们将结果存储在 `r` 中，`r` 通常用来表示估计的相关系数：

```
> r <- cov(launch$temperature, launch$distress_ct) /
      (sd(launch$temperature) * sd(launch$distress_ct))
> r
[1] -0.725671
```

或者，我们可以使用内置的相关系数函数 `cor()`：

```
> cor(launch$temperature, launch$distress_ct)
[1] -0.725671
```

由于相关系数大约为 -0.73，所以这意味着在温度和 O 型环失效数之间存在着相当强的负线性相关性，而负相关意味着温度的升高关联到 O 型环失效数会减少。对于美国国家航空航天局（NASA）的工程师研究的 O 型环数据，这原本是一个非常清晰的指示，即低温发射可能会出问题。

有各种经验规则（大拇指规则）用来解释相关系数。一种方法就是指定相关系数的值在 0.1 ~ 0.3 为弱相关，在 0.3 ~ 0.5 为中相关，超过 0.5 为强相关（这些同样适用于负相关的类似范围）。然而，为了某些目标，这些阈值可能过于宽松。通常，相关性必须根据上下文解释。对于涉及人类的数据，0.5 的相关性可能认为是非常强的；对于机械过程产生的数据，0.5 的相关性可能认为是弱的。



你可能已经听过这种表述：“相关性并不意味着因果性。”这植根于这样的事实，即相关性只描述一对变量之间的关联，但可能还有其他的解释。例如，预期寿命和每天看电影的时间之间可能存在着一种很强的关联性，但是在医生开始建议所有人多看电影之前，我们需要排除另一种解释：老年人看的电影越少，越有可能去世。

度量两个变量之间的相关性给我们提供了一种快速了解因变量和自变量之间关系的方法，当我们开始用大量的预测变量来定义回归模型时，这将变得越来越重要。

6.1.4 多元线性回归

大多数现实世界的分析不止一个自变量。因此，在使用回归来进行数值预测任务时，大多数情况下，很有可能使用多元线性回归（multiple linear regression）。多元线性回归的优点和缺点如下表所示。

优点	缺点
<ul style="list-style-type: none">• 迄今为止，它是数值型数据建模最常用的方法• 可适用于几乎所有的数据• 提供了特征（变量）与结果之间关系的强度与大小的估计	<ul style="list-style-type: none">• 对数据做出了很强的假设• 该模型的形式必须由使用者事先指定• 不能很好地处理缺失数据• 只能处理数值特征，所以分类数据需要额外的处理• 需要一些统计知识来理解模型

我们可以将多元回归作为简单线性回归的扩展来理解。在这两种回归中，目标是相似的：求出系数 β 的值和最大限度减小线性方程的预测误差。主要区别是增加的自变量需要额外的条件。

多元回归方程一般遵循下述方程的形式。因变量 y 是截距项加上每一个特征 i 的 β 估计值与 x 值的乘积。这里已加入误差项（用希腊字母 ε 表示）作为一种提示，即这些预测并不完美，这就是前面所提到的残差项

$$y = \alpha + \beta_1x_1 + \beta_2x_2 + ... + \beta_ix_i + \varepsilon$$

让我们考虑一下估计的回归参数的解释。你会注意到，在前面的方程中，需要对每一个特征估计一个系数，这使得每个特征对 y 的值都有一个单独的估计影响。也就是说，每增加一个单位的 x_i ， y 的变化量为 β_i ，那么截距项就是当所有自变量为零时的 y 的估计值。

由于截距项与任何其他的回归参数相比确实没有什么不同，所以它也可以表示为 β_0 （读作 beta-naught），如下面的方程所示：

$$y = \beta_0 + \beta_1x_1 + \beta_2x_2 + ... + \beta_ix_i + \varepsilon$$

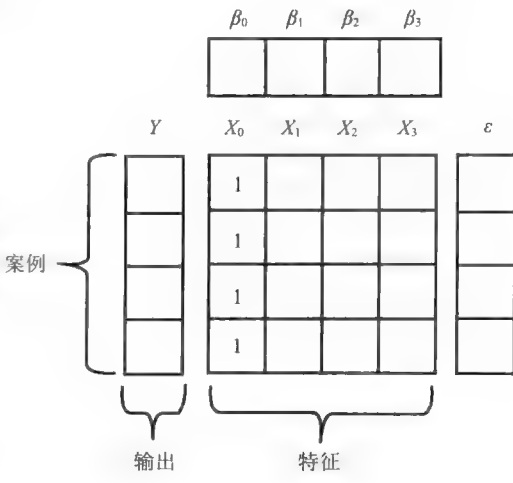
使用简明的公式，该方程可以重新表示为：

$$Y = X\beta + \varepsilon$$

即使该公式看起来很熟悉，但还是有一些细微的变化。因变量现在是一个向量 Y ，一行表示一个案例；自变量被合并成一个矩阵 X ，一列表示一个特征，再加上额外的一列用来表示截距项的值全为“1”的列。同样，回归系数 β 和误差 ε 现在都是向量。

下图说明了这些变化：

现在的目标就是要求出使得 y 的预测值与真实值之间的误差平方和最小的向量 β 。由于寻找最优解需要使用矩阵代数，所以推导过程



比本书中所提供的更需要仔细关注。然而，如果你愿意相信他人的工作，那么向量 β 的最佳估计可以这样计算：

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

该估计值运用了两种矩阵运算： T 表示矩阵 X 的转置，而负指数表示矩阵的逆。使用 R 中内置的矩阵运算，我们可以实现一个简单的多元回归学习，让我们看一看是否可以将该公式应用于挑战者号的发射数据。



如果你不熟悉前面的矩阵运算，维基百科的**转置 (transpose)** 和**矩阵的逆 (matrix inverse)** 页面提供了全面的介绍。而且，即使没有很强的数学背景，也是完全可以理解的。

使用下面的代码，我们可以创建一个名为 `reg` 的简单回归函数，该函数的输入参数为 y 和 x ，并返回一个估计的 β 系数矩阵。

```
> reg <- function(y, x) {
  x <- as.matrix(x)
  x <- cbind(Intercept = 1, x)
  solve(t(x) %*% x) %*% t(x) %*% y
}
```

该函数使用了几个我们之前没有使用过的 R 命令。首先，因为函数要使用来自数据框的列数据，所以需要函数 `as.matrix()` 将数据强制变成矩阵形式。其次，函数 `cbind()` 用来将额外的一列添加到矩阵 x 中，命令 `Intercept = 1` 指示 R 将新的一列命名为 `Intercept`，并将这一列全部用数值 1 填充。最后，关于对象 x 和 y 进行一些矩阵运算：

- 函数 `solve()` 执行矩阵的逆运算。
- 函数 `t()` 用来将矩阵转置。
- `%*%` 将两个矩阵相乘。

如公式所示，结合这些命令来求 β 的估计向量，函数将返回 y 关于 x 的线性模型的估计参数。

让我们将函数 `reg()` 应用于航天飞机的发射数据。如下面的代码所示，数据包含了 4 个特征和一个令人感兴趣的结果 `distress_ct` (O 型环的失效数)：

```
> str(launch)
'data.frame': 23 obs. of 5 variables:
 $ o_ring_ct : int 6 6 6 6 6 6 6 6 6 6 ...
 $ distress_ct: int 0 1 0 0 0 0 0 0 1 1 ...
 $ temperature: int 66 70 69 68 67 72 73 70 57 63 ...
 $ pressure : int 50 50 50 50 50 50 100 100 200 200 ...
 $ launch_id : int 1 2 3 4 5 6 7 8 9 10 ...
```

通过将 `reg()` 函数运行的 O 型环失效数量结果与仅应用温度的简单线性模型的结果相比较，我们发现之前的参数 $a=4.30$ 、 $b=-0.057$ ，于是我们可以证实函数 `reg()` 能正确运行。由于温度位于发射数据的第三列，所以可以如下运行 `reg()` 函数：

```
> reg(y = launch$distress_ct, x = launch[3])
      [,1]
Intercept  4.30158730
temperature -0.05746032
```

这些值与之前的结果完全一样，因此可以使用该函数建立多元回归模型。像之前一样应用该函数，但是这一次，我们将指定 3 列数据而不再只是 1 列数据：

```
> reg(y = launch$distress_ct, x = launch[3:5])
      [,1]
Intercept  3.814247216
temperature -0.055068768
pressure    0.003428843
launch_id  -0.016734090
```

该模型预测 O 型环失效数与温度、压力和发射 ID 号之间的数值关系。温度和发射 ID 号变量的负系数表明，随着温度和发射 ID 号的增加，O 型环失效数会减少。将同样的解释应用于压力变量的系数，我们知道随着压力的增加，预计 O 型环失效数也将增加。



尽管你不是一个科学家，但这些研究结果似乎是合理的。寒冷的气温会使得 O 型环更脆，更大的压力可能会增加零件上的压力，但是为什么发射 ID 号与较少的 O 型环失效数有关呢？一种解释是，也许以后发射使用的 O 型环是由强度更大、更富有弹性的材料构成的。

到目前为止，我们只涉及了线性回归模型的表面。尽管我们的工作确实能够帮助我们理解回归模型是如何建立的，但是 R 中用来拟合线性回归模型的函数不仅比我们的运算更快，而且信息量更丰富。现实世界的回归包提供了大量的输出以帮助解释模型。让我们将我们的回归知识应用到一个更具有挑战性的学习任务中吧！

6.2 例子——应用线性回归预测医疗费用

公司为了赚钱，保险需要募集比花费在受益者的医疗服务上更多的年度保费，因此，保险公司投入了大量的时间和金钱来研发能精确预测医疗费用的模型。

医疗费用很难估计，因为花费最高的情况是罕见的而且似乎是随机的。但是有些情况对于部分特定的群体还是比较普遍存在的。例如，吸烟者比不吸烟者得肺癌的可能性更大，肥胖的人更有可能得心脏病。

此分析的目的是利用病人的数据来预测这部分群体的平均医疗费用。这些估计可以用来创建一个精算表，根据预期的治疗费用来设定年度保费价格是高一点还是低一点。

6.2.1 第 1 步——收集数据

为了便于分析，我们使用一个模拟数据集，该数据集包含了美国病人的医疗费用。而本书创建的这些数据使用了来自美国人口普查局（U.S. Census Bureau）的人口统计资料，因此

可以大致反映现实世界的情况。



如果你想一起学习这个例子，那么你需要从 Packt 出版社的网站下载 `insurance.csv` 文件，并将该文件保存到 R 的工作文件夹中。

该文件 (`insurance.csv`) 包含 1338 个案例，即目前已经登记过的保险计划受益者以及表示病人特点和历年计划计入的总的医疗费用的特征。这些特征是：

- `age`：这是一个整数，表示主要受益者的年龄（不包括超过 64 岁的人，因为他们一般由政府支付）。
- `sex`：这是保单持有人的性别，要么是 `male`，要么是 `female`。
- `bmi`：这是**身体质量指数**（Body Mass Index, BMI），它提供了一个判断人的体重相对于身高是过重还是偏轻的方法，BMI 指数等于体重（公斤）除以身高（米）的平方。一个理想的 BMI 指数在 18.5 ~ 24.9 的范围内。
- `children`：这是一个整数，表示保险计划中所包括的孩子 / 受抚养者的数量。
- `smoker`：根据被保险人是否吸烟判断 `yes` 或者 `no`。
- `region`：根据受益人在美国的居住地，分为 4 个地理区域：`northeast`、`southeast`、`southwest` 和 `northwest`。

如何将这些变量与已结算的医疗费用联系在一起是非常重要的。例如，我们可能认为老年人和吸烟者在大额医疗费用上有较高的风险。与许多其他的机器学习方法不同，在回归分析中，特征之间的关系通常由使用者指定而不是自动检测出来。在下一节中，我们将探讨其中的一些潜在关系。

6.2.2 第 2 步——探索和准备数据

正如我们以前所做的那样，我们将使用 `read.csv()` 函数来加载用于分析的数据。我们可以安全地使用 `stringsAsFactors = TRUE`，因为将名义变量转换成因子变量是恰当的：

```
> insurance <- read.csv("insurance.csv", stringsAsFactors = TRUE)
```

函数 `str()` 确认该数据转换了我们之前所期望的形式：

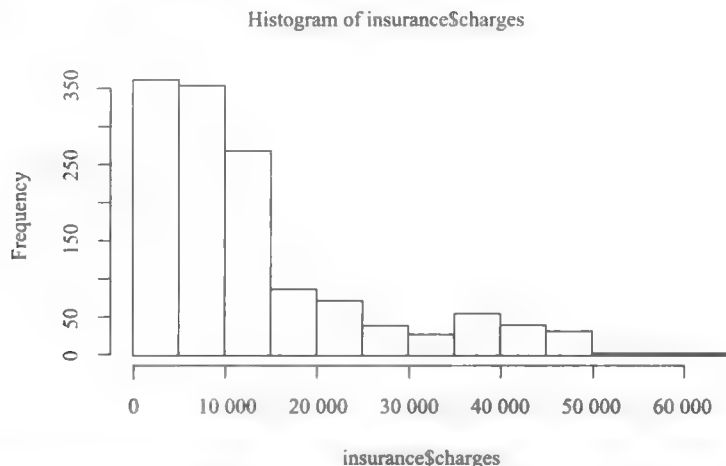
```
> str(insurance)
'data.frame': 1338 obs. of 7 variables:
 $ age      : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex      : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 ...
 $ bmi      : num  27.9 33.8 33 22.7 28.9 ...
 $ children : int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker   : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 ...
 $ region   : Factor w/ 4 levels "northeast","northwest", ...
 $ charges  : num  16885 1726 4449 21984 3867 ...
```

既然因变量是 `charges`，那么让我们一起来看一看它是如何分布的：

```
> summary(insurance$charges)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1122   4740   9382  13270  16640  63770
```

因为平均值远大于中位数，所以这表明保险费用的分布是右偏的。我们可以使用直观的直方图来证实这一点。

```
> hist(insurance$charges)
```



在我们的数据中，绝大多数的个人每年的医疗费用都在 0 ~ 15 000 美元，尽管分布的尾部经过直方图的峰部后延伸得很远。因为线性回归假设因变量的分布为正态分布，所以这种分布是不理想的。在实际应用中，线性回归的假设往往会被违背。如果需要，我们在后面能够修正该假设。

即将面临的另一个问题就是回归模型需要每一个特征都是数值型的，而在我们的数据框中，我们有 3 个因子类型的特征。很快，我们就会看到 R 中的线性回归函数如何处理我们的变量。

变量 `sex` 被划分成 `male` 和 `female` 两个水平，而变量 `smoker` 被划分成 `yes` 和 `no` 两个水平。从 `summary()` 的输出中，我们知道变量 `region` 有 4 个水平，但我们需要仔细看一看，它们是如何分布的。

```
> table(insurance$region)
northeast northwest southeast southwest
      324         325         364         325
```

这里，我们看到数据几乎均匀地分布在 4 个地理区域中。

1. 探索特征之间的关系——相关系数矩阵

在使用回归模型拟合数据之前，有必要确定自变量与因变量之间以及自变量之间是如何相关的。**相关系数矩阵 (correlation matrix)** 提供了这些关系的快速概览。给定一组变量，它可以为每一对变量之间的关系提供一个相关系数。

为 `insurance` 数据框中的 4 个数值型变量创建一个相关系数矩阵，可以使用 `cor()` 命令：

```
> cor(insurance[c("age", "bmi", "children", "charges")])
      age      bmi  children  charges
age  1.0000000 0.1092719 0.0424690 0.29900819
bmi   0.1092719 1.0000000 0.0127589 0.19834097
children 0.0424690 0.0127589 1.0000000 0.06799823
charges 0.2990082 0.1983410 0.06799823 1.00000000
```

在每个行与列的交叉点，列出的相关系数表示其所在的行与其所在的列的两个变量之间的相关系数。对角线始终为 1，因为一个变量和其自身之间总是完全相关的。因为相关性是对称的，换句话说就是， $\text{cor}(x, y) = \text{cor}(y, x)$ ，所以对角线上方的值与其下方的值是相同的。

该矩阵中的相关系数不是强相关的，但还是存在一些显著的关联。例如，`age` 和 `bmi` 显示出中度相关，这意味着随着年龄 (`age`) 的增长，身体质量指数 (`bmi`) 也会增加。此外，`age` 和 `charges`，`bmi` 和 `charges`，以及 `children` 和 `charges` 也都呈现出中度相关。当我们建立最终的回归模型时，我们会尽量更加清晰地梳理出这些关系。

2. 可视化特征之间的关系——散点图矩阵

或许通过使用散点图，可视化特征之间的关系会更有帮助。虽然我们可以为每个可能的关系创建一个散点图，但对于大量的特征，这样做可能会变得比较烦琐。

另一种方法就是创建一个散点图矩阵 (scatterplot matrix) (有时简称为 **SPLOM**)，就是简单地将一个散点图集合排列在网格中，它可以用来检测 3 个或者更多变量之间的模式，但散点图矩阵并不是真正的多维可视化，因为一次只能研究两个特征。尽管如此，它还是提供了一种通用的研究数据是如何内在相关的方法。

我们可以使用 R 中的图形功能来为 4 个数值型的特征 (`age`、`bmi`、`children` 和 `charges`) 创建一个散点图矩阵。默认的 R 安装中就提供了函数 `pairs()`，该函数为产生散点图矩阵提供了基本的功能。为了调用该函数，只需要给它提供数据框，结果就呈现出散点图矩阵。这里，我们将把 `insurance` 数据框限制为感兴趣的 4 个数值型变量：

```
> pairs(insurance[c("age", "bmi", "children", "charges")])
```

这样就产生了下面的图：

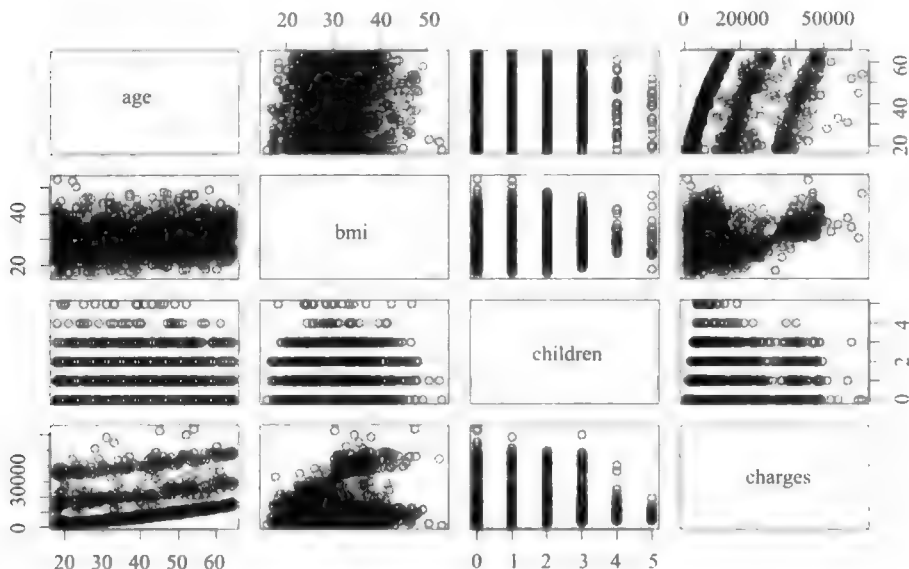
与相关系数矩阵一样，每个行与列的交叉点所在的散点图表示其所在的行与列的两个变量的相关关系。由于对角线上方和下方的 x 轴和 y 轴是交换的，所以对角线上方的图和下方的图是互为转置的。

你注意到这些散点图中的一些图案了吗？尽管有一些看上去像是随机密布的点，但还是有一些似乎呈现了某种趋势。`age` 和 `charges` 之间的关系呈现出几条相对的直线，而 `bmi` 和 `charges` 的散点构成了两个不同的群体。在其他任何散点图中都很难检测出趋势。

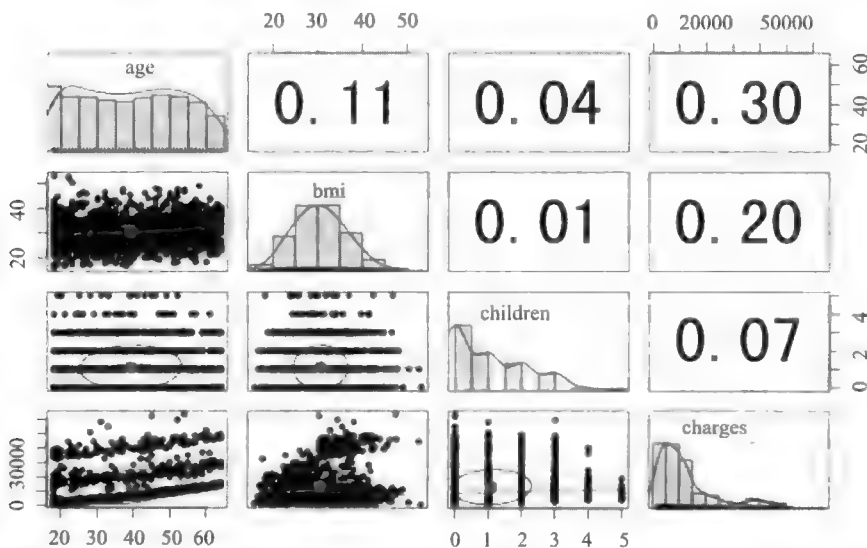
如果我们对散点图添加更多的信息，那么它就会更加有用。一个改进后的散点图矩阵可以用 `psych` 包中的 `pairs.panels()` 函数来创建。如果你还没有安装这个包，那

么可以输入 `install.packages("psych")` 命令将其安装到你的系统中，并使用 `library(psych)` 命令加载它。然后，就可以像之前所做的那样创建一个散点图矩阵：

```
> pairs.panels(insurance[c("age", "bmi", "children", "charges")])
```



这将产生一个略微丰富的散点图矩阵，如下图所示



在对角线的上方，散点图被相关系数矩阵所取代。在对角线上，直方图描绘了每个特征的数值分布。最后，对角线下方的散点图带有额外的可视化信息。

每个散点图中呈椭圆形的对象称为**相关椭圆** (correlation ellipse)，它提供了一种变量之间是如何密切相关的可视化信息。位于椭圆中心的点表示 x 轴变量的均值和 y 轴变量的均值所确定的点。两个变量之间的相关性由椭圆的形状所表示，椭圆越被拉伸，其相关性越强。一个几乎类似于圆的完美的椭圆形，如 `bmi` 和 `children`，表示一种非常弱的相关性（在这种情况下相关系数为 0.01）。

散点图中绘制的曲线称为**局部回归平滑** (loess smooth)，它表示 x 轴和 y 轴变量之间的一般关系。最好通过例子来理解。散点图中 `age` 和 `children` 的曲线是一个倒置的 U，峰值在中年附近，这意味着案例中年龄最大的人和年龄最小的人比年龄大约在中年附近的人拥有的孩子要少。因为这种趋势是非线性的，所以这一发现已经不能单独从相关性推断出来。另一方面，对于 `age` 和 `bmi`，局部回归平滑是一条倾斜的逐渐上升的线，这表明 BMI 会随着年龄 (`age`) 的增长而增加，但我们已经从相关系数矩阵中推断出该结论。

6.2.3 第 3 步——基于数据训练模型

用 R 对数据拟合一个线性回归模型，可以使用 `lm()` 函数。该函数包含在 `stats` 添加包中，当你安装 R 软件时，该包就应该默认安装并在 R 启动时自动加载好了。函数 `lm()` 的语法如下所示：

下面命令拟合称为 `ins_model` 的线性回归模型，该模型将 6 个自变量与总的医疗费用联系在一起。R 中的公式语法使用波浪号 `~` 来描述模型，因变量位于波浪号的左侧，自变量位于波浪号的右侧，自变量通过符号 `+` 隔开。这里没有必要指定回归模型的截距项，因为在默认的情况下，它是假定存在的：

```
> ins_model <- lm(charges ~ age + children + bmi + sex +
smoker + region, data = insurance)
```

多元回归模型语法

应用 `stats` 添加包中的函数 `lm()`

建立模型：

```
m <- lm(dv ~ iv, data = mydata)
```

- `dv`：是 `mydata` 数据框中需要建模的因变量
- `iv`：为一个 R 公式，用来指定 `mydata` 数据框中被用于模型的自变量
- `data`：为包含变量 `dv` 和变量 `iv` 的数据框

该函数返回一个回归模型对象，该对象能够用于预测。自变量之间的交互作用可以通过运算符 `*` 来给出

进行预测：

```
p <- predict(m, test)
```

- `m`：由函数 `lm()` 训练的一个模型
 - `test`：一个包含测试数据的数据框，该数据框和用来建立模型的训练数据有同样的特征
- 该函数将返回一个含有预测值的向量。

例子：

```
ins_model <- lm(charges ~ age + children + sex + smoke, rdata = insurance)
ins_pred <- predict(ins_model, insurance_test)
```

由于符号 `.` 可以用来指定所有的特征（不包括那些公式中已经指定的），所以下面的命令等价于前面的命令：

```
> ins_model <- lm(charges ~ ., data = insurance)

建立模型后，只需输入该模型对象的名称，就可以看到估计的  $\beta$  系数：
```

```
> ins_model3

Call:
lm(formula = charges ~ age + children + bmi + sex +
    smoker + region, data = insurance)

Coefficients:
      (Intercept)          age        children
        -11938.5         256.9          475.5

        bmi        sexmale        smokeryes
         339.2        -131.3         23848.5

regionnorthwest regionsoutheast regionsouthwest
        -353.0         -1035.0         -960.1
```

理解回归系数是相当简单的。截距告诉我们当自变量的值都等于 0 时 `charges` 的值。



与这里的案例一样，截距往往是很难解释的，因为使所有特征的取值都为 0 是不可能的。例如，因为没有人的年龄（`age`）和 BMI 是取值为 0 的，所以截距没有内在的意义。出于这个原因，截距在实际中常常被忽略。

在其他特征保持不变时，一个特征的 β 系数表示该特征每增加一个单位，`charges`（费用）的增加量。例如，随着每一年年龄的增加，假设其他一切都一样（不变），我们将会预计平均增加 \$256.90 的医疗费用。同样，每增加一个孩子，每年将会产生平均 \$475.50 的额外医疗费用；而每增加一个单位的 BMI，每年的医疗费用将会增加 \$339.20。

你可能注意到，虽然在我们的模型公式中，我们仅指定了 6 个变量，但是输出时，除了截距项之外，却输出了 8 个系数。之所以发生这种情况，是因为 `lm()` 函数自动将一种称为虚拟编码（dummy coding）的技术应用于模型所包含的每一个因子类型的变量中。

虚拟编码允许名义特征通过为一个特征的每一类创建一个二元变量来将其处理成数值型变量，即如果观测值属于某一类，那就设定为 1，否则设定为 0。例如，性别（`sex`）变量有两类：男性（`male`）和女性（`female`）。这将分为两个二进制值变量，R 中将其命名为 `sexmale` 和 `sexfemale`。对于观测值，如果 `sex = male`，那么 `sexmale = 1`、`sexfemale = 0`；如果 `sex = female`，那么 `sexmale = 0`、`sexfemale = 1`。相同的编码适用于有 3 个类别甚至更多类别的变量，具有 4 个类别的特征 `region` 可以分为 4 个变量：`regionnorthwest`、`regionsoutheast`、`regionsouthwest` 和 `regionnortheast`。

当添加一个虚拟编码的变量到回归模型中时，一个类别总是被排除在外作为参照类别。然后，估计的系数就是相对于参照类别解释的。在我们的模型中，R 自动保留 `sexfemale`、

smokerno 和 regionnortheast 变量,使东北地区的女性非吸烟者作为参照组。因此,相对于女性来说,男性每年的医疗费用要少 \$131.30;吸烟者平均多花费 \$23 848.50,远超过非吸烟者。此外,模型中另外 3 个地区的系数是负的,这意味着东北地区倾向于具有最高的平均医疗费用。



默认情况下,R 中使用 factor (因子) 变量的第一个水平作为参照组。如果你想使用另一类(水平),那么可以使用 relevel() 函数来手动指定参照组,使用 R 中的 ?relevel 命令可获取更多信息。

线性回归模型的结果是合乎逻辑的。高龄、吸烟和肥胖往往与其他健康问题联系在一起,而额外的家庭成员或者受抚养者可能会导致就诊次数增加和预防保健(比如接种疫苗、每年体检)费用的增加。然而,我们目前并不知道该模型对数据的拟合有多好?我们将在下一节中回答这个问题。

6.2.4 第 4 步——评估模型的性能

通过在 R 命令行中输入 ins_model,可以获得参数的估计值,它们告诉我们关于自变量是如何与因变量相关联的。但是它们根本没有告诉我们用该模型来拟合数据有多好。为了评估模型的性能,可以使用 summary() 命令来分析所存储的回归模型:

```
> summary(ins_model)
```

这就产生了以下的输出:

```
Call:
lm(formula = charges ~ age + children + bmi + sex + smoker +
    region, data = insurance)

Residuals:
    Min       1Q   Median       3Q      Max    1
-11304.9 -2848.1  -982.1   1393.9 29992.8

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    2
(Intercept)  -11938.5      987.8  -12.086 < 2e-16 ***
age             256.9       11.9   21.587 < 2e-16 ***
children       475.5       137.8    3.451 0.000577 ***
bmi            339.2       28.6   11.860 < 2e-16 ***
sexmale       -131.3       332.9   -0.394 0.693348
smokeryes     23848.5      413.1   57.723 < 2e-16 ***
regionnorthwest -353.0       476.3   -0.741 0.458769
regionsoutheast -1035.0      478.7   -2.162 0.030782 *
regionsouthwest -960.0       477.9   -2.009 0.044765 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1

Residual standard error: 6062 on 1329 degrees of freedom
Multiple R-squared:  0.7509,    Adjusted R-squared:  0.7494    3
F-statistic: 500.8 on 8 and 1329 DF,  p-value: < 2.2e-16
```

开始时,summary() 的输出可能看起来令人费解,但基本原理是很容易掌握的。与上述输出中用标签编号所表示的一样,该输出为评估模型的性能提供了 3 个关键的方面:

1) Residuals (残差) 部分提供了预测误差的主要统计量,其中有一些统计量显然是相当大的。由于残差等于真实值减去预测值,所以最大误差值 29 992.8 表明该模型至少对一个

案例的费用少预测了近 \$30 000。另一方面，误差值的 50% 落在 1Q 值到 3Q 值内（第一四分位数和第三四分位数），所以大部分的预测值在超过真实值 \$2 850 与低于真实值 \$1 400 之间。

2) 星号（例如，***）表示模型中每个特征的预测能力：给定估计值，显著性水平（正如图输出下方的“Signif.codes”部分所列出的）提供了一种度量方式，即真实系数有多大可能性为 0。其中，3 颗星的出现表示显著性水平为 0，这意味着该特征极不可能与因变量无关的变量，而一个通常的做法就是使用 0.05 的显著性水平来表示统计意义上的显著变量。如果模型中几乎没有特征在统计意义上是显著的，那么这可能会引起关注，因为这将表明特征是不能预测结果的。这里，模型有几个显著的变量，而且从逻辑上，它们看起来是与结果相关的。

3) 多元 R 方值（也称为判定系数）提供了一种度量模型性能的方式，即从整体上，模型能多大程度解释因变量的值。它类似于相关系数，因为它的值越接近于 1.0，模型解释数据的性能就越好。由于 R 方值为 0.749，所以我们知道，近 75% 的因变量的变化程度可以由模型解释。因为模型的特征越多，模型解释的变化程度就越大，所以调整 R 方值通过惩罚具有很多自变量的模型来修正 R 方值，用它来比较具有不同数目的解释变量的模型的性能是很有用的。

给定前面 3 个性能指标，我们的模型表现得相当好。对于现实世界数据的回归模型，R 方值相当低的情况并不少见，因此 0.75 的 R 方值实际上是相当不错的。考虑到医疗费用的性质，其中有些误差的大小是需要关注的，但并不令人吃惊。然而，如下一节所述，我们可能会以略微不同的方式来指定模型，从而提高模型的性能。

6.2.5 第 5 步——提高模型的性能

正如前面所提到的，回归模型和其他机器学习方法的一个关键区别就在于回归通常会让使用者来选择特征和设定模型。因此，如果我们有关于一个特征是如何与结果相关的学科知识，我们就可以使用该信息来对模型进行设定，并可能提高模型的性能。

1. 模型的设定——添加非线性关系

在线性回归中，自变量和因变量之间的关系被假定为是线性的，然而这不一定是正确的。例如，对所有的年龄值来讲，年龄对医疗费用的影响可能不是恒定的；对于最老的人群，治疗可能会过于昂贵。

如果你还记得，一个典型的回归方程遵循如下的类似形式：

$$y = \alpha + \beta_1 x$$

考虑到非线性关系，可以添加一个高阶项到回归模型中，把模型当做多项式处理。实际上，我们将建立一个如下所示的关系模型：

$$y = \alpha + \beta_1 x + \beta_2 x^2$$

这两个模型之间的区别在于将估计一个单独的 β_2 ，其目的是捕捉 x^2 项的效果，这允许通

过一个年龄的平方项来度量年龄的影响。

为了将非线性年龄添加到模型中，我们只需要创建一个新的变量：

```
> insurance$age2 <- insurance$age^2
```

然后，当我们建立改进后的模型时，我们将把 age 和 age2 都添加到 lm() 公式中，例如，`charges ~ age + age2`。

2. 转换——将一个数值型变量转换为一个二进制指标

假设我们有一种预感，一个特征的影响不是累积的，而是当特征的取值达到一个给定的阈值后才产生影响。例如，对于在正常体重范围内的个人来说，BMI 对医疗费用的影响可能为 0，但是对于肥胖者（即 BMI 不低于 30）来说，它可能与较高的费用密切相关。

我们可以通过创建一个二进制指标变量来建立这种关系，即如果 BMI 大于等于 30，那么设定为 1，否则设定为 0。该二元特征的 β 估计表示 BMI 大于等于 30 的个人相对于 BMI 小于 30 的个人对医疗费用的平均净影响。

要创建一个特征，我们可以使用 `ifelse()` 函数，该函数用于对向量中的每一个元素测试一个指定的条件，并根据条件是为 true 还是 false，返回一个值。对于 BMI 大于等于 30，我们将返回 1，否则返回 0：

```
> insurance$bmi30 <- ifelse(insurance$bmi >= 30, 1, 0)
```

然后，可以在改进的模型中包含 bmi30 变量，无论是取代原来的 bmi 变量，还是作为补充，这取决于我们是否认为除了一个单独的 BMI 影响外，肥胖的影响也会发生。如果没有很好的理由不这样做，那么我们将在最终的模型中包含两者。



如果你在决定是否要包含一个变量时遇到困难，一种常见的做法就是包含它并检验其显著性水平。然后，如果该变量在统计上不显著，那么就有证据支持在将来排除该变量。

3. 模型的设定——加入相互作用的影响

到目前为止，我们只考虑了每个特征对结果的单独影响（贡献）。如果某些特征对因变量有综合影响，那么该怎么办呢？例如，吸烟和肥胖可能分别都有有害的影响，但是假设它们的共同影响可能会比它们每一个单独影响之和更糟糕是合理的。

当两个特征存在共同的影响时，这称为**相互作用**（interaction）。如果怀疑两个变量相互作用，那么可以通过在模型中添加它们的相互作用来检验这一假设，可以使用 R 中的公式语法来指定相互作用的影响。为了体现肥胖指标（bmi30）和吸烟指标（smoker）的相互作用，可以这样的形式写一个公式：`charges ~ bmi30*smoker`。

运算符 `*` 是一个简写，用来指示 R 对如下进行建模：

```
charges ~ bmi30 + smokeryes + bmi30:smokeryes
```

在展开式中，运算符：(冒号)表示 `bmi30:smokeryes` 是两个变量之间的相互作用。注意，该展开式还自动包括 `bmi30` 和 `smoker` 变量及其相互作用。



如果模型中没有添加每一个相互作用的变量，那么相互作用就不应该包含在模型中。如果你总是使用运算符 * 创建相互作用，那么这将不是一个问题，因为 R 将自动添加所需要的变量。

4. 全部放在一起——一个改进的回归模型

基于医疗费用如何与患者特点联系在一起的一点学科知识，我们开发了一个我们认为更加精确专用的回归公式。下面就总结一下我们的改进：

- ❑ 增加了一个非线性年龄项。
- ❑ 为肥胖创建了一个指标。
- ❑ 指定了肥胖和吸烟之间的相互作用。

我们将像之前一样使用 `lm()` 函数来训练模型，但是这一次，我们将添加新构造的变量和相互作用项：

```
> ins_model2 <- lm(charges ~ age + age2 + children + bmi + sex +
  bmi30*smoker + region, data = insurance)
```

接下来，我们概述结果：

```
> summary(ins_model2)
```

```
Call:
lm(formula = charges ~ age + age2 + children + bmi + sex + bmi30 *
  smoker + region, data = insurance)

Residuals:
    Min       1Q   Median       3Q      Max
-17296.4  -1656.0  -1263.3   -722.1   24160.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   134.2509    1362.7511    0.099  0.921539
age           -32.6851     59.8242   -0.546  0.584915
age2             3.7316     0.7463    5.000  6.50e-07 ***
children       678.5612    105.8831    6.409  2.04e-10 ***
bmi           120.0196     34.2660    3.503  0.000476 ***
sexmale       -496.8245    244.3659   -2.033  0.042240 *
bmi30        -1000.1403    422.8402   -2.365  0.018159 *
smokeryes     13404.6866    439.9491   30.469  < 2e-16 ***
regionnorthwest -279.2038     349.2746   -0.799  0.424212
regionsoutheast -828.5467     351.6352   -2.356  0.018604 *
regionsouthwest -1222.6437     350.5285   -3.488  0.000503 ***
bmi30:smokeryes 19810.7533    604.6567   32.764  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4445 on 1326 degrees of freedom
Multiple R-squared:  0.8664, Adjusted R-squared:  0.8653
F-statistic: 781.7 on 11 and 1326 DF, p-value: < 2.2e-16
```

分析该模型的拟合统计量有助于确定我们的改变是否提高了回归模型的性能。相对于我们的第一个模型，R 方值从 0.75 提高到约 0.87，我们的模型现在能解释医疗费用变化的 87%。此外，我们关于模型函数形式的理论似乎得到了验证，高阶项 `age2` 在统计上是显著

的, 肥胖指标 $bmi30$ 也是显著的。肥胖和吸烟之间的相互作用表明了一个巨大的影响, 除了单独吸烟增加的超过 \$13 404 的费用外, 肥胖的吸烟者每年要另外花费 \$19 810, 这可能表明吸烟会加剧 (恶化) 与肥胖有关的疾病。

6.3 理解回归树和模型树

你是否还记得在第 5 章中, 决策树建立了一个很像流程图的模型, 在这个模型中, 决策节点、叶节点和分支定义了一系列可用于案例分类的决策。通过对树的生长算法做小幅调整, 这些树可以同样应用于数值预测。在本节中, 我们将只考虑决策树用于数值预测, 不同于将其用于分类。

决策树用于数值预测可分为两类。第一类称为**回归树** (regression tree), 是在 20 世纪 80 年代作为**分类回归树** (Classification and Regression Tree, CART) 算法的一部分引入的。尽管叫这个名字, 但是正如本章前面所描述的, 回归树并没有使用线性回归的方法, 而是基于到达叶节点的案例的平均值做出预测。



CART 算法的详细信息参见 *Classification and Regression Trees* by L. Breiman, J.H. Friedman, C.J. Stone, and R.A. Olshen (Chapman & Hall, 1984)。

用于数值预测的第二类决策树称为**模型树** (model trees), 比回归树要晚几年引入, 它们不太广为人知, 但或许功能更强大。模型树和回归树以大致相同的方式生长, 但是在每个叶节点, 根据到达该节点的案例建立多元线性回归模型。根据叶节点的数目, 一棵模型树可能会建立几十甚至几百个这样的模型, 这可能会使模型树比同等的回归树更难理解, 但好处是它们也许能建立一个更加精确的模型。



最早的模型树算法 M5 的描述参见 *Learning with Continuous Classes*, Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343-348, by J.R. Quinlan (1992)。

将回归加入到决策树

能够进行数值预测的决策树提供了一个引人注目, 但经常被忽略的, 可以取代回归模型的方法。相对于更常见的回归方法, 回归树和模型树的优点与缺点都列在下表中。

优点	缺点
<ul style="list-style-type: none"> 将决策树的优点与对数值型数据建模的能力相结合 能自动选择特征, 允许该方法与大量特征一起使用 不需要使用者事先指定模型 拟合某些类型的数据可能会比线性回归好得多 不要求用统计的知识来解释模型 	<ul style="list-style-type: none"> 不像线性回归那样常用 需要大量的训练数据 难以确定单个特征对于结果的总体净影响 可能比回归模型更难解释

虽然传统的回归方法通常是数值预测任务的第一选择，但是在某些情况下，数值决策树提供了明显的优势。例如，决策树可能更适合于具有许多特征或者特征和结果之间具有许多复杂、非线性关系的任务，这些情形给回归带来了挑战。而且，回归建模关于数值型数据是如何分布的假设往往被现实世界的的数据所违背，但决策树就不存在这样的情况。

用于数值预测的决策树的建立方式与用于分类的决策树的建立方式大致相同。从根节点开始，按照特征使用分而治之的策略对数据进行划分，在进行一次分割后，将会导致结果最大化地均匀增长。在分类决策树中，一致性（均匀性）是由熵值来度量的，而对于数值型数据是未定义的。于是，对于数值决策树，一致性（均匀性）可以通过统计量（比如方差、标准差或者平均绝对偏差）来度量。根据所使用的决策树生长算法，一致性（均匀性）度量可能会有所不同，但原理是基本相同的。

一个常见的分割标准称为**标准偏差减少**（Standard Deviation Reduction, SDR），它由下面的公式定义：

$$\text{SDR} = \text{sd}(T) - \sum_i \frac{|T_i|}{|T|} \times \text{sd}(T_i)$$

在这个公式中，函数 $\text{sd}(T)$ 指的是集合 T 中的值的标准差，而 T_1, T_2, \dots, T_n 是由对于一个特征的一次分割产生的值的集合； $|T|$ 项指的是集合 T 中观测值的数量。从本质上讲，公式度量的是从原始值的标准差减去分割后加权的标准差的减少量。

举个例子，考虑下面的情况，其中，一棵决策树需要决定是对二元特征 A 进行分割还是对二元特征 B 进行分割。

原始数据	1	1	1	2	2	3	4	5	5	6	6	7	7	7	7
根据特征A划分	1	1	1	2	2	3	4	5	5	6	6	7	7	7	7
根据特征B划分	1	1	1	2	2	3	4	5	5	6	6	7	7	7	7
	T_1							T_2							

使用由图中所建议的分割产生的组，可以如下对 A 和 B 计算 SDR，这里使用 `length()` 函数返回一个向量中元素的数目。注意，整个组 T 命名为 `tee`，以避免覆盖 R 中内置的 `T()` 和 `t()` 函数。

```
> tee <- c(1, 1, 1, 2, 2, 3, 4, 5, 5, 6, 6, 7, 7, 7, 7)
> at1 <- c(1, 1, 1, 2, 2, 3, 4, 5, 5)
> at2 <- c(6, 6, 7, 7, 7, 7)
> bt1 <- c(1, 1, 1, 2, 2, 3, 4)
> bt2 <- c(5, 5, 6, 6, 7, 7, 7, 7)
> sdr_a <- sd(tee) - (length(at1) / length(tee) * sd(at1) +
+                     length(at2) / length(tee) * sd(at2))
> sdr_b <- sd(tee) - (length(bt1) / length(tee) * sd(bt1) +
+                     length(bt2) / length(tee) * sd(bt2))
```

让我们来比较 A 和 B 的 SDR：

```
> sdr_a
[1] 1.202815
> sdr_b
[1] 1.392751
```

关于 A 的分割的 SDR 值大约为 1.2，关于 B 的分割的 SDR 值大约为 1.4。由于对特征 B ，标准差减少得更多，所以决策树将首先使用特征 B ，它产生了比特征 A 略多的一致性（均匀性）集合。

假设决策树在这里停止生长，只使用了这一个分割，那么回归树的工作就完成了。它可以为新的案例进行预测，取决于它们是落入组 T_1 还是组 T_2 。如果案例最后在 T_1 中，那么模型将预测 $\text{mean}(\text{bt1})=2$ ，否则将预测 $\text{mean}(\text{bt2})=6.25$ 。

相比之下，模型树将多走一步。使用落入组 bt1 的 7 个训练案例和落入组 bt2 的 8 个训练案例，模型树可以建立一个结果相对于特征 A 的线性回归模型（在回归模型中，特征 B 没有任何帮助，因为所有位于叶节点的个案与 B 有相同的值）。然后，模型树可以使用两个线性模型中的任何一个为新的案例做出预测。

为了进一步说明这两种方法之间的差异，我们研究一个现实世界的例子。

6.4 例子——用回归树和模型树估计葡萄酒的质量

葡萄酒酿造是一个充满挑战和竞争力的行业，它为巨大的利润提供了可能。然而，也有诸多因素有助于提升一个葡萄酒酿造厂的盈利能力。作为一种农产品，有包括天气和生长环境在内的多个变量影响用特定品种葡萄酿造的酒的质量。装瓶和生产同样会影响风味，是更好还是更差。甚至产品进入市场的方式，从瓶身的设计到零售价，都会影响顾客的味道感。

因此，葡萄酒酿造业已经在数据采集和可能有助于葡萄酒酿造决策科学的机器学习方法中投入了巨资。例如，机器学习已经用来发现来自不同地区的葡萄酒化学成分的主要差异，或者用来确定导致葡萄酒味道更甜的化学因素。

近来，机器学习已经用来协助葡萄酒质量的评级——一个极其困难的任务。由一位知名的葡萄酒评论家撰写的一份评论往往决定了该产品最终是在货架的顶部还是底部，尽管事实上在双盲试验中对葡萄酒评级时，专家评委的意见是不一致的。

在这个案例学习中，我们将使用回归树和模型树来创建一个能模仿葡萄酒专家评级的系统。由于决策树产生的模型很容易理解，所以这可以让葡萄酒酿造师来确定有助于葡萄酒更好评级的关键因素。或许更重要的是，该系统不需要忍受品酒的人为因素，比如评级者的情绪和鉴赏疲劳。因此，计算机辅助的葡萄酒测试可能会产生更好的产品以及更客观、一致、公平的评级。

6.4.1 第 1 步——收集数据

为了研究葡萄酒评级模型，我们将使用由 P. Cortez, A. Cerdeira、F. Almeida、T. Matos 和 J. Reis 捐赠给 UCI 机器学习数据仓库（UCI Machine Learning Data Repository）的数据。这

些数据包括来自葡萄牙（世界领先的葡萄酒生产国之一）的红色和白色 Vinho Verde（青酒）葡萄酒案例。因为有助于获得高度评价的葡萄酒的因素可能在红色和白色品种之间有所不同，所以为了便于分析，我们将只研究较受欢迎的白葡萄酒。



要理解这个例子，你需要从 Packt 出版社的网站下载 `whitewines.csv` 文件，并将该文件保存到 R 的工作目录下。如果你想自己研究这些数据，文件 `redwines.csv` 也可以下载。

白葡萄酒数据包含了 4898 个葡萄酒案例的 11 种化学特性的信息。对于每种葡萄酒，实验室分析测量的特性包括酸性、含糖量、氯化物含量、硫的含量、酒精度、pH 值和密度。然后，这些样本会由不少于 3 名鉴定者组成的小组以盲品的方式进行评级，质量尺度从 0（很差）到 10（极好）。如果鉴定者对于评级没有达成一致意见，那么就会使用中间值。

Cortez 的研究评估了 3 种机器学习方法多元回归、人工神经网络和支持向量机对葡萄酒数据建立模型的能力。在本章的前面，我们已经介绍了多元回归，我们将在第 7 章学习神经网络和支持向量机。该研究发现，支持向量机提供了比线性回归模型显著更好的结果。然而，与回归不同的是，支持向量机模型很难解释。使用回归树和模型树，我们或许能够改善回归的结果，同时还能拥有一个容易理解的模型。



要了解更多关于葡萄酒的研究，请参考已发表的文章：Modeling wine preferences by data mining from physicochemical properties, *Decision Support Systems*, Vol. 47, pp. 547-553, by P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis (2009)。

6.4.2 第 2 步——探索和准备数据

通常，使用 `read.csv()` 函数将数据加载到 R 中。由于所有的特征都是数值型的，所以我们可以放心地忽略 `stringsAsFactors` 参数。

```
> wine <- read.csv("whitewines.csv")
```

wine（葡萄酒）数据包括 11 个特征和结果变量 `quality`（品质结果），如下所示：

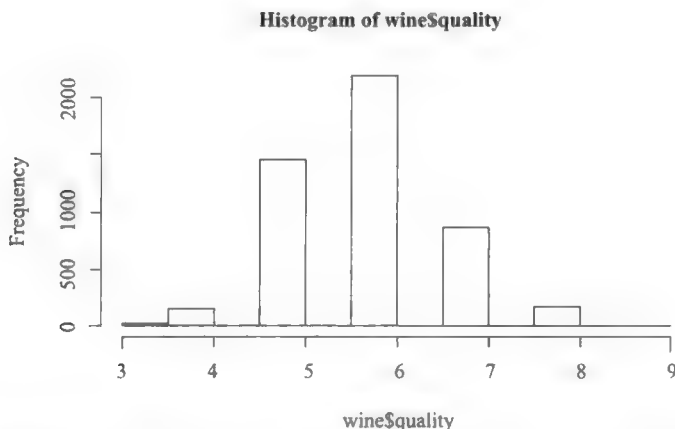
```
> str(wine)
'data.frame': 4898 obs. of 12 variables:
 $ fixed.acidity      : num  6.7 5.7 5.9 5.3 6.4 7 7.9 ...
 $ volatile.acidity   : num  0.62 0.22 0.19 0.47 0.29 0.12 ...
 $ citric.acid        : num  0.24 0.2 0.26 0.1 0.21 0.41 ...
 $ residual.sugar     : num  1.1 16 7.4 1.3 9.65 0.9 ...
 $ chlorides          : num  0.039 0.044 0.034 0.036 0.041 ...
 $ free.sulfur.dioxide : num  6 41 33 11 36 22 33 17 34 40 ...
 $ total.sulfur.dioxide : num  62 113 123 74 119 95 152 ...
 $ density            : num  0.993 0.999 0.995 0.991 0.993 ...
 $ pH                 : num  3.41 3.22 3.49 3.48 2.99 3.25 ...
 $ sulphates          : num  0.32 0.46 0.42 0.54 0.34 0.43 ...
 $ alcohol            : num  10.4 8.9 10.1 11.2 10.9 ...
 $ quality            : int  5 6 6 4 6 6 6 6 6 7 ...
```

相比于其他类型的机器学习模型，决策树的优点之一就是它们可以处理多种类型的数据而无需进行预处理。这意味着不需要将特征规范化或者标准化。

然而，为了增加模型评价的信息，还是需要花点精力来研究结果变量分布的。例如，假设葡萄酒之间在质量上几乎没有变化，或者葡萄酒落入一个双峰分布：要么非常好，要么非常差。这些情况可能会给模型带来麻烦。为了检查这种极端情形，我们可以使用直方图来研究葡萄酒质量的分布：

```
> hist(wine$quality)
```

这就产生了下面的图：



葡萄酒的质量值似乎遵循一个相当于正态的钟形分布，大约以数值 6 为中心。从直观上看，这是有意义的，因为大部分葡萄酒的质量为平均质量，少数葡萄酒特别差或者特别好。尽管这里没有显示结果，但是研究 `summary(wine)` 的输出同样有益于发现异常值或者其他潜在的数据问题。虽然决策树对于难以处理的数据是相当稳健的，但是它总会谨慎地检查严重的问题。目前，我们将假设数据是可靠的。

然后，最后一步就是将数据分为训练数据集和测试数据集。由于 `wine`（葡萄酒）数据已经处理成随机的顺序，所以可以将数据分割成两个连续行的集合，如下所示：

```
> wine_train <- wine[1:3750, ]
> wine_test <- wine[3751:4898, ]
```

为了反映 Cortez 使用过的条件，我们分别使用 75% 的数据集用于训练，25% 的数据集用于测试。我们将根据测试数据来评估基于决策树的模型的性能，并看看我们是否能够获得与先前的研究学习相媲美的结果。

6.4.3 第 3 步——基于数据训练模型

我们将从训练一个回归树模型开始。虽然几乎决策树的所有实现都可以用来进行回归树建模，但是 `rpart`（递归划分）添加包中提供了像 CART（分类回归树）团队中所描述的

最可靠的回归树实现。作为 CART 的经典 R 实现，`rpart` 添加包同样有着充分的帮助文档，有用于可视化和评估 `rpart` 模型的多个函数的支持。

安装 `rpart` 添加包需要使用 `install.packages("rpart")` 命令。然后，使用命令 `library(rpart)`，就可以将其加载到 R 的会话中。使用下面的语法，所包含的 `rpart()` 函数就可以拟合分类树或者回归树。该函数将使用默认的设置来拟合一棵决策树，通常情况下，其执行效果相当好。如果你需要更多的微调设置，请使用 `?rpart.control` 命令来了解控制参数。

回归树语法
应用 <code>rpart</code> 添加包中的函数 <code>rpart()</code>
<p>建立模型：</p> <pre>m <- rpart(dv ~ iv, data = mydata)</pre> <ul style="list-style-type: none"> • <code>dv</code>: 是 <code>mydata</code> 数据框中需要建模的因变量 • <code>iv</code>: 为一个 R 公式，用来指定 <code>mydata</code> 数据框中被用于模型的自变量 • <code>data</code>: 为包含变量 <code>dv</code> 和变量 <code>iv</code> 的数据框 <p>该函数返回一个回归树模型对象，该对象能够用于预测</p> <p>进行预测：</p> <pre>p <- predict(m, test, type = "vector")</pre> <ul style="list-style-type: none"> • <code>m</code>: 由函数 <code>rpart()</code> 训练的一个模型 • <code>test</code>: 一个包含测试数据的数据框，该数据框和用来建立模型的训练数据有同样的特征 • <code>type</code>: 给定返回的预测值的类型，取值为 <code>"vector"</code> (预测数值型数据)，或者 <code>"class"</code> (预测类别)，或者 <code>"prob"</code> (预测类别的概率)。 <p>该函数的返回值取决于 <code>type</code> 参数，它是一个含有预测值的向量。</p> <p>例子：</p> <pre>wine_model <- rpart(quality ~ alcohol + sulfates, data = wine_train) wine_predictions <- predict(wine_model, wine_test)</pre>

使用 R 的公式界面，我们可以指定 `quality` (质量) 为结果变量 (因变量)，并使用点符号 `"."` 使得 `wine_train` (葡萄酒训练) 数据中的其他所有列被用来作为预测变量 (自变量)。由此产生的模型对象命名为 `m.rpart`，以区分后面将要训练的模型树：

```
> m.rpart <- rpart(quality ~ ., data = wine_train)
```

获取关于该树的基本信息，只需要输入该模型对象的名称：

```
> m.rpart
n= 3750
```

```
node), split, n, deviance, yval
* denotes terminal node
```

```
1) root 3750 2945.53200 5.870933
 2) alcohol < 10.85 2372 1418.86100 5.604975
   4) volatile.acidity >= 0.2275 1611 821.30730 5.432030
```

```

8) volatile.acidity>=0.3025 688 278.97670 5.255814 *
9) volatile.acidity< 0.3025 923 505.04230 5.563380 *
5) volatile.acidity< 0.2275 761 447.36400 5.971091 *
3) alcohol>=10.85 1378 1070.08200 6.328737
6) free.sulfur.dioxide< 10.5 84 95.55952 5.369048 *
7) free.sulfur.dioxide>=10.5 1294 892.13600 6.391036
14) alcohol< 11.76667 629 430.11130 6.173291
28) volatile.acidity>=0.465 11 10.72727 4.545455 *
29) volatile.acidity< 0.465 618 389.71680 6.202265 *
15) alcohol>=11.76667 665 403.99400 6.596992 *

```

对于决策树中的每个节点，到达决策点的案例数量都列出来了。例如，所有的 3750 个案例从根节点开始，其中，有 2372 个案例的 `alcohol < 10.85`，1378 个案例的 `alcohol >= 10.85`。因为 `alcohol`（酒精）是决策树中第一个使用的变量，所以它是葡萄酒质量中唯一最重要的指标。

用 * 表示的节点是终端或者叶节点，这意味着它们会产生预测（这里作为 `yval` 列出来）。例如，节点 5 有一个 5.971 091 的 `yval`。当该决策树用来预测时，对任意一个葡萄酒案例，如果其 `alcohol < 10.85` 且 `volatile.acidity < 0.2275`，那么它的质量值将预测为 5.97。

关于该决策树拟合的更详细的总结，包括每一个节点的均方误差和整体特征重要性的度量，可以通过使用 `summary(m.rpart)` 命令获得。

可视化决策树

尽管只使用前面的输出就可以理解该决策树，但是使用可视化通常更容易理解。由 Stephen Milborrow 创建的 `rpart.plot` 包提供了一个易于使用的函数来生成具有出版质量的决策树。



关于 `rpart.plot` 更多的信息，包括该函数可以生成的其他样本类型的决策树图形，请参考该作者的网站：<http://www.milbo.org/rpart-plot/>。

在使用 `install.packages("rpart.plot")` 命令安装该包后，`rpart.plot()` 函数可以根据任意一个 `rpart` 模型对象生成一个决策树图形。下面的命令绘制了我们之前建立的回归树的图：

```

> library(rpart.plot)
> rpart.plot(m.rpart, digits = 3)

```

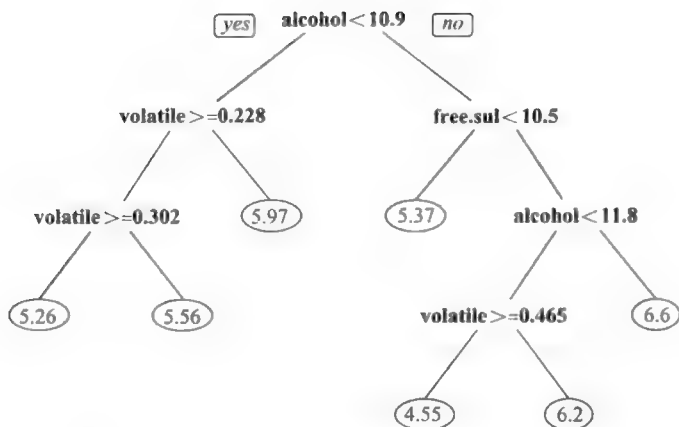
产生的决策树图形如下所示。

除了用于控制包含在图中数字位数的参数 `digits` 以外，许多可视化的其他方面都可以调整。下面的命令仅显示了几个有用的选项，参数 `fallen.leaves` 强制叶节点与图的底部保持一致（对齐），而参数 `type` 和参数 `extra` 影响决策和节点被标记的方式。

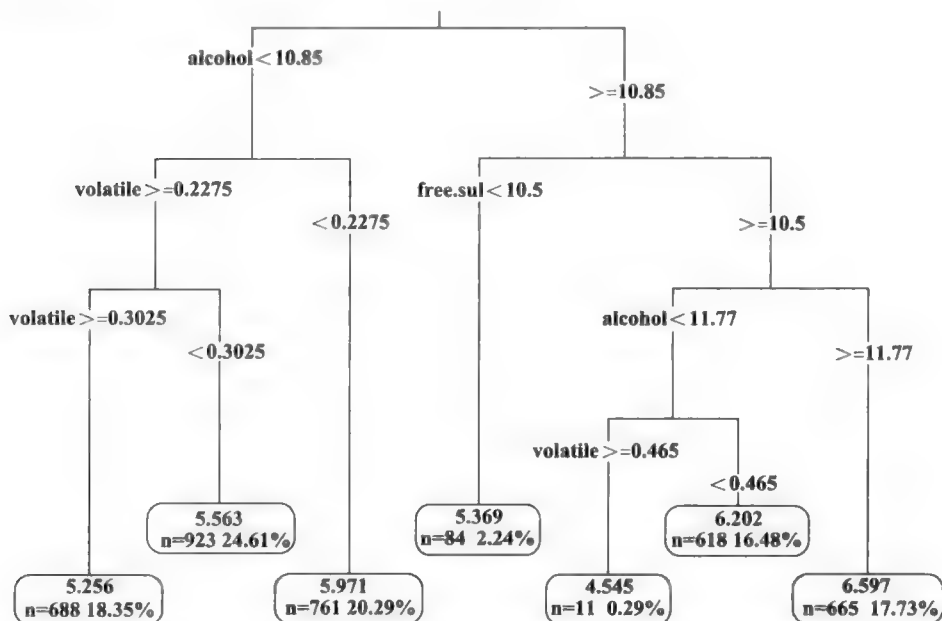
```

> rpart.plot(m.rpart, digits = 4, fallen.leaves = TRUE,
type = 3, extra = 101)

```



这些变化的结果是一个看上去截然不同的树形图：



6.4.4 第4步——评估模型的性能

为了使用基于测试数据的回归树模型进行预测，我们使用 `predict()` 函数。在默认情况下，该函数返回结果变量的数值估计，我们将把它的返回值保存在一个名为 `p.rpart` 的向量中：

```
> p.rpart <- predict(m.rpart, wine_test)
```

我们预测的主要统计量表明了一个潜在的问题，预测值与真实值相比落在一个更窄的范围内：

```
> summary(p.rpart)
```



```

      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
4.545    5.563    5.971    5.893    6.202    6.597
> summary(wine_test$quality)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
3.000    5.000    6.000    5.901    6.000    9.000

```

这一发现表明，该模型不能正确识别极端的情形，尤其是最好的和最差的葡萄酒。另一方面，在第一四分位数和第三四分位数分位数之间，我们可能做得不错。

预测的质量（quality）值和真实的质量（quality）值之间的相关性提供了一种度量模型性能的简单方法。回想一下，cor() 函数可以用来度量两个相同长度向量之间的关系，使用该函数可以比较预测值对应于真实值的程度有多好：

```

> cor(p.rpart, wine_test$quality)
[1] 0.5369525

```

相关系数 = 0.54 肯定是可以接受的。然而，相关系数只是度量了预测值与真实值的相关性有多强，而不是度量预测值离真实值有多远的方法。

用平均绝对误差度量性能

一般来说，另一种思考模型性能的方法就是考虑它的预测值离真实值有多远，这种度量方法称为平均绝对误差（Mean Absolute Error, MAE）。MAE 的方程如下所示，其中， n 表示预测值的数量， e_i 表示第 i 个预测值的误差：

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |e_i|$$

从本质上讲，这个方程得到的是误差绝对值的均值。由于误差仅仅是预测值与真实值之间的差值，所以可以创建一个简单的 MAE() 函数，如下所示：

```

> MAE <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

```

然后，预测的 MAE 为：

```

> MAE(p.rpart, wine_test$quality)
[1] 0.5872652

```

就平均而言，这意味着模型的预测值与真实的质量分数之间的差值大约为 0.59。基于质量的尺度是从 0 ~ 10，这似乎表明我们的模型做得相当好。

另一方面，回忆一下，大多数葡萄酒既不是很好也不是很差。通常情况下，质量的分数大约为 5 ~ 6。因此，根据这个指标，一个什么都没有做而是仅仅预测了均值的分类器可能同样会做得相当好。

训练数据中的平均质量等级如下所示：

```

> mean(wine_train$quality)
[1] 5.870933

```

如果我们对每一个葡萄酒案例预测的值为 5.87，那么我们将只有大约 0.67 的平均绝对误差：

```
> mean_abserror(5.87, wine_test$quality)
[1] 0.6722474
```

回归树 ($MAE = 0.59$) 比估算的均值 ($MAE = 0.67$) 平均更接近于真实的质量分数，但相差不是很大。作为比较，Cortez 报告了神经网络模型的 MAE 为 0.58，支持向量机的 MAE 为 0.45。这表明，模型还有改善的空间。

6.4.5 第 5 步——提高模型的性能

为了提高学习算法的性能，我们尝试构建一棵模型树。回想一下，模型树可以通过回归模型取代叶节点来改善回归树。这通常会导致比回归树更精确的结果，而回归树在叶节点进行预测时只使用了一个单一的值。

目前，模型树中最先进的算法是由 Wang and Witten 提出的 **M5' 算法** (M5-prime)，这是对 Quinlan 在 1992 年提出的原始 M5 模型树算法的一个改进。



关于 M5' 算法的更多信息，请参考：Induction of model trees for predicting continuous classes, Proceedings of the Poster Papers of the European Conference on Machine Learning by Y. Wang and I.H. Witten (1997)。

M5' 算法在 R 中通过 RWeka 包和 M5P() 函数可以得到。该函数的语法如下表所示。如果你尚未安装 RWeka 添加包，请确保安装该添加包。由于其依赖于 Java，所以安装的说明都包含在第 1 章中。

模型树语法
应用 RWeka 添加包中的函数 M5P()
<p>建立模型：</p> <pre>m <- M5P(dv ~ iv, data = mydata)</pre> <ul style="list-style-type: none"> ● dv: 是 mydata 数据框中需要建模的因变量 ● iv: 为一个 R 公式，用来指定 mydata 数据框中被用于模型的自变量 ● data: 为包含变量 dv 和变量 iv 的数据框 <p>该函数返回一个模型树对象，该对象能够用于预测</p> <p>进行预测：</p> <pre>p <- predict(m, test)</pre> <ul style="list-style-type: none"> ● m: 由函数 M5P() 训练的一个模型 ● test: 一个包含测试数据的数据框，该数据框和用来建立模型的训练数据有同样的特征 <p>该函数返回一个含有预测值的数值向量。</p> <p>例子：</p> <pre>wine_model <- M5P(quality ~ alcohol + sulfates, data = wine_train) wine_predictions <- predict(wine_model, wine_test)</pre>

我们将使用和回归树基本相同的语法来拟合模型树：

```
> library(RWeka)
> m.m5p <- M5P(quality ~ ., data = wine_train)
```

树本身可以通过输入它的名称来查看。在这种情况下，树是很大的，只显示了输出的前几行：

```
> m.m5p
M5 pruned model tree:
(using smoothed linear models)
alcohol <= 10.85 :
|   volatile.acidity <= 0.238 :
|   |   fixed.acidity <= 6.85 : LM1 (406/66.024%)
|   |   fixed.acidity > 6.85 :
|   |   |   free.sulfur.dioxide <= 24.5 : LM2 (113/87.697%)
```

你将会注意到分割与我们前面建立的回归树很相似。酒精（alcohol）是最重要的变量，紧接着是挥发性酸（volatile acidity）和游离二氧化硫（free sulfur dioxide）。然而，一个关键的区别在于节点不是以一个数值预测终止，而是以一个线性模型终止（这里表示为 LM1 和 LM2）。

线性模型本身显示在输出的后面。例如，LM1 模型如下所示。这些值完全可以像我们在本章前面建立的多元回归模型一样解释，每一个数字都是相关的特征对于预测的葡萄酒质量的净影响（效应）。对于固定酸度的系数 0.266 意味着每增加一个单位的酸度，葡萄酒的质量预计会增加 0.266。

```
LM num: 1
quality =
  0.266 * fixed.acidity
- 2.3082 * volatile.acidity
- 0.012 * citric.acid
+ 0.0421 * residual.sugar
+ 0.1126 * chlorides
+ 0 * free.sulfur.dioxide
- 0.0015 * total.sulfur.dioxide
- 109.8813 * density
+ 0.035 * pH
+ 1.4122 * sulphates
- 0.0046 * alcohol
+ 113.1021
```

值得注意的是，估计的影响只适用于到达该节点的葡萄酒案例。在这个模型树中一共建立了 36 个线性模型，每一个模型对于固定酸度和其他 10 个特征的影响都有不同的估计。

关于模型对训练数据拟合程度好坏的统计量，summary() 函数可应用于 M5P 模型。然而，请记住，因为这些统计量是基于训练数据，所以它们只能作为一个粗略的诊断使用：

```
> summary(m.m5p)

=== Summary ===

Correlation coefficient          0.6666
```

```

Mean absolute error          0.5151
Root mean squared error      0.6614
Relative absolute error      76.4921 %
Root relative squared error   74.6259 %
Total Number of Instances    3750

```

相反，我们将观察基于未知的测试数据模型的性能有多好。`predict()` 函数为我们获取了一个预测值向量：

```
> p.m5p <- predict(m.m5p, wine_test)
```

似乎模型树的预测值范围比回归树更广：

```

> summary(p.m5p)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 4.389   5.430   5.863   5.874   6.305   7.437

```

相关性似乎也有大幅提高：

```

> cor(p.m5p, wine_test$quality)
[1] 0.6272973

```

此外，该模型的平均绝对误差略有改善：

```

> MAE(wine_test$quality, p.m5p)
[1] 0.5463023

```

尽管我们没有以很大的提高来超越回归树，但是我们超越了由 Cortez 发表的神经网络模型的性能，而且我们也更接近了由支持向量机模型发布的 0.45 的平均绝对误差值，同时还使用了一个更简单的学习方法。



不足为奇的是，我们已经证实了预测葡萄酒的质量是一个困难的问题，毕竟品酒本质上是主观的。如果你想要更多的实践，那么你可以在阅读第 11 章后，重温这个问题，因为第 11 章介绍了更多的技巧，可能会带来更好的结果。

6.5 总结

在本章中，呈现了对数值型数据建模的两种方法：第一种方法，线性回归，涉及用直线拟合数据；第二种方法，使用决策树进行数值预测。后者有两种形式：一是回归树，它使用位于叶节点的案例均值进行数值预测；二是模型树，它以一种混合的方法在每一个叶节点建立一个回归模型，该混合方法在某些方面是两个模型中最好的。

采用线性回归模型为不同阶层的人群计算了预期医疗费用。因为特征和变量之间的关系可以用所估计的回归模型描述，所以我们能够确认某些人口统计数据，比如吸烟者和肥胖者，可能需要以要价更高的保险率来支付高于平均水平的医疗费用。

回归树和模型树被用来根据葡萄酒可测量特性，对葡萄酒的主观质量进行建模。在此过程中，我们学习了回归树如何提供一种简单的方法来解释特征和数值结果之间的关系，但是更复杂的模型树可能会更精确。此外，在上述过程中，我们也学到了几种方法来估计数值模型的性能。

本章介绍的机器学习方法让我们对于输入和输出之间的关系有一个清晰的理解，与本章形成鲜明对比的是，下一章介绍的方法产生近乎无法理解的模型，但优势是它们有极其强大的技巧（最强大的分类器之一），它们既可以应用于分类预测问题，也可以应用于数值预测问题。

黑箱方法——神经网络和支持向量机

已故科幻作家阿瑟·克拉克（Arthur C. Clarke）曾经写道：“任何足够先进的技术都是与魔法难以区分的。”本章介绍两个机器学习方法，同样，它们可能第一眼看上去似乎是魔法。作为最强大的机器学习算法中的两个，它们的应用遍及许多领域中的任务。然而，它们的内部运作却很难理解。

在工程中，这些称为**黑箱**（black box）过程，因为将输入转换成输出的机制是通过一个黑盒子来模糊处理的。晦涩难懂（不透明）的原因有诸多方面，例如，封闭源码软件的黑箱故意隐瞒专有算法；制作香肠的黑箱有意模糊化（但好吃）；政治立法的黑箱植根于官僚的做事过程。在机器学习的情况下，黑箱是因为潜在的模型基于复杂的数学系统，而且结果难以解释。

尽管黑箱模型的解释是不可行的，但是盲目应用这些方法是有危险的。因此，在本章中，我们将窥探一些算法背后的知识，并调查涉及拟合这些模型的统计过程。你将会学到：

- ❑ 为了模拟任意函数（功能），神经网络借用了人们理解人脑所应用的一些概念
- ❑ 支持向量机使用多维曲面来定义特征和结果的关系。
- ❑ 尽管它们很复杂，但是这些模型可以很容易地应用到现实世界的问题中，比如模拟混凝土的强度或者阅读印刷文字。

运气好的话，你将会意识到，在统计中，你并不需要具备黑带资格来应对黑箱机器学习方法——没有必要被吓到！

7.1 理解神经网络

人工神经网络（Artificial Neural Network，ANN）对一组输入信号和一组输出信号之间的

关系进行建模，使用的模型来源于人类大脑对来自感觉输入的刺激是如何反应的理解。就像大脑使用一个称为**神经元**（neuron）的相互连接的细胞网络来创建一个巨大的并行处理器一样，人工神经网络使用人工神经元或者**节点**（node）的网络来解决学习问题。

人脑大约由 850 亿个神经元构成，产生了一个能够存储巨量知识的网络。正如你可能期望的，这使得其他生物的大脑相形见绌。例如，一只猫大约有 10 亿个神经元，一只老鼠大约有 7500 万个神经元，一只蟑螂大约只有 100 万个神经元。相比之下，许多人工神经网络包含的神经元要少得多，通常只有几百个，所以我们在不久的将来随时创建一个人工大脑是没有危险的——即使是一只具有 10 万个神经元的果蝇也远远超过了目前最先进的人工神经网络。



虽然神经网络可能不适合用来完全模拟一只蟑螂的大脑，但是它可能提供一个关于蟑螂行为的充分的探索模型，比如在一个算法中，它可以模拟当一只蟑螂被发现时，它是如何逃离的。如果一只蟑螂的行为是令人信服的，那么它的大脑是如何运作的重要吗？这个问题是具有争议的**图灵测试**（Turing test）的基础，即如果一个人不能将机器的行为与一种生物的行为区分开来，那么图灵测试将该机器划分为智能类。

基本的人工神经网络的历史已超过 50 年，它们通过模拟大脑的方法来解决问题。最初，这涉及学习简单的函数，如逻辑 AND 函数或者逻辑 OR。这些早期的练习主要用于构建生物大脑可能会如何起作用的模型。然而，近年来，随着计算机的功能变得越来越强大，人工神经网络的复杂性也同样增加了，使得它们现在经常应用于更实际的问题，比如：

- ❑ 如那些使用语音信箱转接服务和邮政信件分拣机的语音和字迹识别程序
- ❑ 如一座办公楼的环境控制或者自动驾驶的汽车和无人机的智能设备的自动化
- ❑ 天气和气候模式，拉伸强度，流体动力学，许多其他科学，社会和经济现象的复杂模型。

从广义上讲，人工神经网络是可以应用于几乎所有学习任务的多功能学习方法：分类、数值预测，甚至无监督的模式识别。

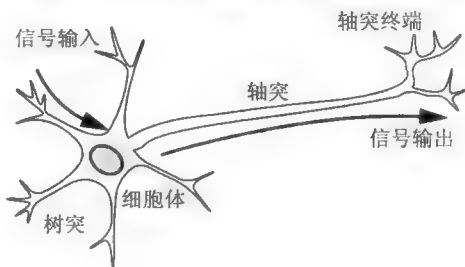


不管值得与否，人工神经网络学习方法经常在媒体上大张旗鼓地报道。例如，一个由谷歌开发的“人工大脑”因为其具有能够识别 You Tube（世界上最大的视频分享网站）上猫的视频的能力而被吹捧。这样的炒作与人工神经网络的任何独特性关系很小，却与人工神经网络很有魅力的事实有很大关系，因为它们与生物的大脑很相似。

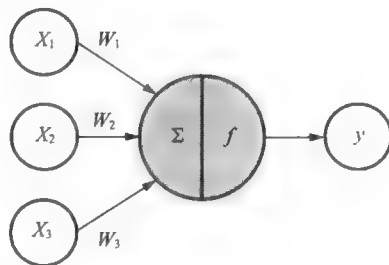
人工神经网络最好应用于下列问题：输入数据和输出数据都很好理解或者至少相当简单，但其涉及输入到输出的过程是极其复杂的。作为一种黑箱方法，对于这些类型的黑箱问题，它们运行得很好。

7.1.1 从生物神经元到人工神经元

由于人工神经网络故意设计为人脑活动的概念模型，所以首先理解生物神经元如何发挥作用是有帮助的。如下图所示，细胞的**树突**（dendrite）通过一个允许神经冲动根据其相对重要性或者频率加权的生化过程来接受输入的信号。由于细胞体开始积累输入信号，所以当达到一个阈值之后，细胞便会充满活力击破，然后输出信号通过一个电化过程传送到**轴突**（axon）。在轴突终末，该电信号会再次被处理为一种化学信号，穿过称为**突触**（synapse）的一个微小间隙传递到相邻的神经元。



一个单一的人工神经元模型可以用非常类似于生物模型的术语来理解。如下图所示，一个有向网络图定义了树突接收的输入信号（变量 x ）和输出信号（变量 y ）之间的关系。与生物神经元一样，每一个树突的信号都根据其重要性被加权（ w 值）——现在先忽略如何确定这些权重。输入信号由细胞体求和，然后该信号根据一个用 f 表示的**激活函数**（activation function）来传递。



一个典型的有 n 个输入树突的神经元可以用下面的公式表示。权重 w 可以控制 n 个输入（ x ）中的每个输入对输入信号之和所做的贡献的大小。激活函数 $f(x)$ 使用净总和，结果信号 $y(x)$ 就是输出轴突。

$$y(x) = f\left(\sum_{i=1}^n w_i x_i\right)$$

就像使用积木那样，神经网络应用这种方式定义的神经元来构建复杂的数据模型。虽然有很多种不同的神经网络，但是每一种都可以由下面的特征来定义：

- **激活函数**（activation function），将神经元的净输入信号转换成单一的输出信号，以便进一步在网络中传播。
- **网络拓扑**（network topology）（或结构），描述了模型中神经元的数量以及层数和它们连接的方式。
- **训练算法**（training algorithm），指定如何设置连接权重，以便抑制或者增加神经元在输入信号中的比重。

让我们来看看上述每一种特征的不同情况，看看如何用它们来构建典型的神经网络模型。

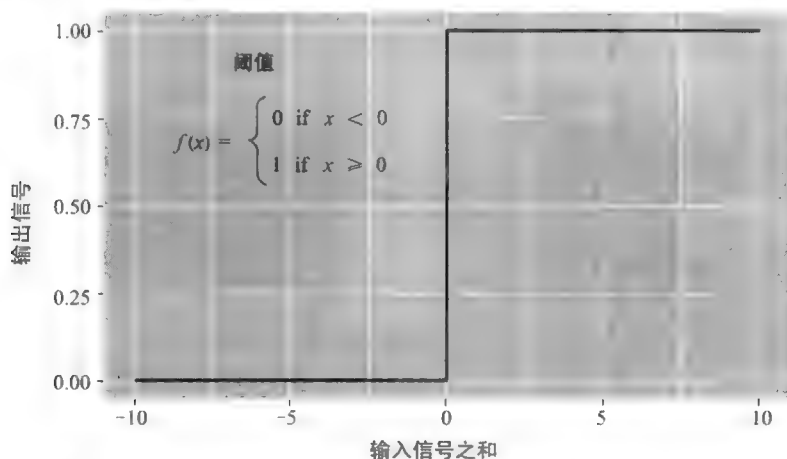
7.1.2 激活函数

激活函数是人工神经元处理信息并将信息传递到整个网络的机制。正如人工神经元是以

生物中的神经元为模型，激活函数也是以生物神经元的机制为模型。

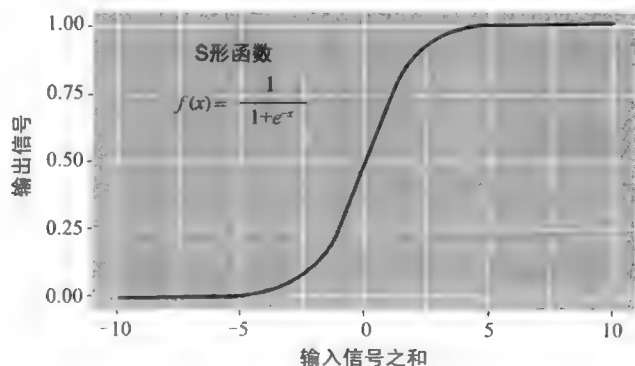
在生物世界中，激活函数可以想象为一个过程，涉及对总的输入信号求和，和确定其是否满足激活阈值。如果满足，神经元传递信号；否则，它不执行任何操作。在人工神经网络术语中，这称为**阈值激活函数**（threshold activation function），因为它仅在一个指定的输入阈值达到后，才产生一个输出信号。

下图显示了一个典型的阈值函数。在这种情况下，当输入信号的总和至少为零时，神经元才击破阈值。因为它的形状，有时它也称为**单位跳跃激活函数**（unit step activation function）。

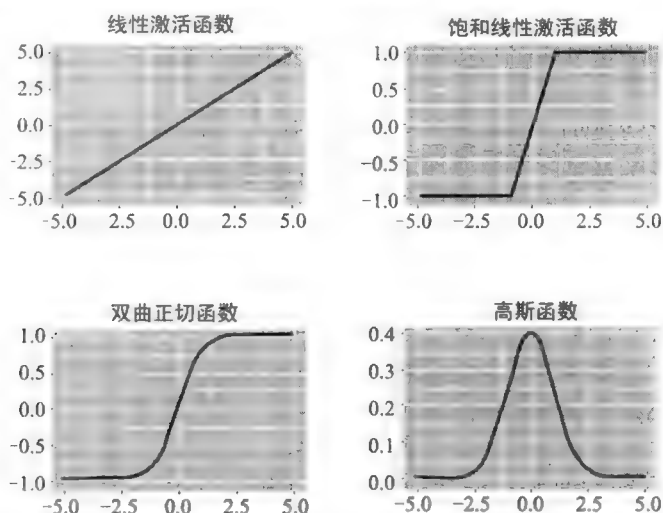


虽然该激活函数因为其与生物学的相似之处很有趣，但是它很少用于人工神经网络中。从生物化学的局限中解脱出来，根据它们解释令人满意的数学特征的能力和对数据之间的关系建立模型的能力来选择人工神经网络激活函数。

或许最常用的方法是下图所示的**S形激活函数**（sigmoid activation function）（特别是逻辑S形，the logistic sigmoid），其中e是自然对数的底（约为2.72）。尽管它与阈值激活函数有类似的步骤或者S的形状，但是输出信号不再是二元的，输出值可以落在0 ~ 1的任何地方。此外，该S形激活函数是**可微的**，这意味着它很有可能计算出遍及整个输入范围的导数。正如你后面将要学习的，该特征对于创建高效的人工神经网络优化算法是至关重要的。



虽然该 S 形激活函数也许是最常用的激活函数，并且通常在默认情况下使用，但是有些神经网络允许选择其他的激活函数。这样的激活函数的选择如下图所示。



构成这些激活函数之间差异的主要细节就是输出信号的范围不同。通常情况下，输出信号范围是 $(0, 1)$ 、 $(-1, +1)$ 或者 $(-\infty, +\infty)$ 的一种。激活函数的选择与具体的神经网络有关，它可能更适合拟合某些类型的数据，允许构建专门的神经网络。例如，线性激活函数产生非常类似于线性回归模型的神经网络，而高斯激活函数产生称为**径向基函数 (Radial Basis Function, RBF)** 网络模型。

重要的是要认识到，对于许多激活函数，影响输出信号的输入值范围是相对较窄的。例如，在 S 形激活函数情况下，对于一个低于 -5 或者超过 +5 的输入信号，输出信号始终为 0 或者 1。这种方式的信号压缩会导致一个饱和信号位于非常动态化的输入的最高端或者最低端，就像把一把吉他的扩音器调到很高以致由于削减声波的峰值而使声音失真。因为本质上这是将输入值压缩到一个较小的输出范围，所以这样的激活函数（像 S 形）有时候称为**压缩函数 (squashing function)**。

压缩问题的解决方法就是转换所有的神经网络输入，使特征值落在 0 附近的小范围内。通常情况下，这可以通过标准化或者规范化完成。通过限制输入值，激活函数将对整个范围采取行动，从而预防取值尺度很大的特征（例如，收入）支配取值尺度较小的特征（例如，家庭的孩子个数）。另一个好处是，这些模型也可能更快地训练，因为这些算法可以通过输入值的可操作范围更快地迭代。



虽然理论上一个神经网络可以经过多次迭代来调整它的权重以适应非常动态化的特征，但是在极端情况下，许多算法将在此发生之前就已经停止了迭代。如果你的模型预测是没有意义的，请检查你是否已经正确标准化了输入数据。

7.1.3 网络拓扑

神经网络的学习能力来源于它的**拓扑结构** (topology), 或者相互连接的神经元的模式与结构。虽然有无数的网络结构形式, 但是它们可以通过 3 个关键特征来区分:

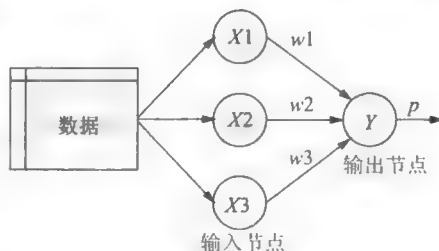
- 层的数目。
- 网络中的信息是否允许向后传播。
- 网络中每一层内的节点数。

拓扑结构决定了可以通过网络进行学习任务的复杂性。一般来说, 更大、更复杂的网络能够识别更微妙的模式及更复杂的决策边界。然而, 神经网络的效能不仅是一个网络规模的函数, 也取决于其构成元素的组织方式。

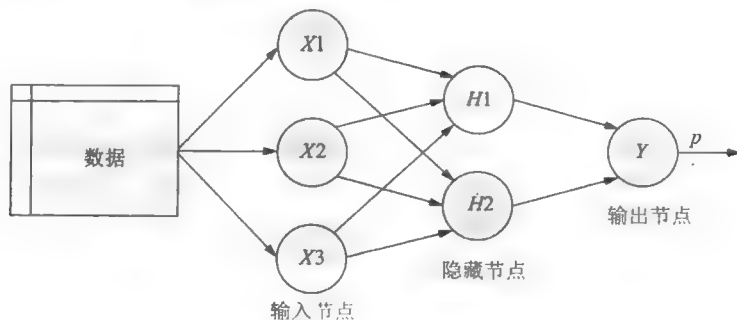
1. 层的数目

为了定义拓扑, 需要一个术语来区分位于网络中不同位置的人工神经元。下图显示了一个非常简单网络的拓扑结构。称为**输入节点** (Input Node) 的一组神经元直接从输入的数据接收未经处理的信号, 然后每个输入节点负责处理数据集中一个单一的特征, 该特征的值将由节点的激活函数进行转换。从输入节点产生的信号由**输出节点** (Output Node) 接收, 输出节点使用它自己的激活函数来生成最终的预测 (这里记为 P)。

输入节点和输出节点安排在称为**层** (layer) 的组中。因为输入节点处理正确接收的输入数据, 所以该网络只有一组连接权重 (这里标记为 w_1 、 w_2 和 w_3)。因此它称为**单层网络** (single-layer network)。单层网络可以用于基本的模式分类, 特别是可用于能够线性分割的模式, 但大多数的学习任务需要更复杂的网络。



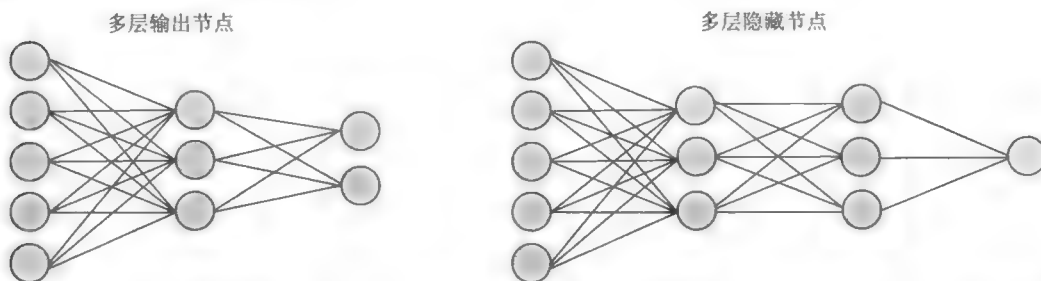
正如你可能所期望的, 一种明显的创建更复杂网络的方法是通过添加额外的层。就像这里所描绘的, **多层网络** (multilayer network) 添加了一个或者更多的**隐藏层** (hidden layer), 它们在信号到达输出节点之前处理来自输入节点的信号。大多数多层网络被**完全连接** (fully connected), 这意味着前一层中的每个节点都连接到下一层中的每个节点, 但这不是必需的。



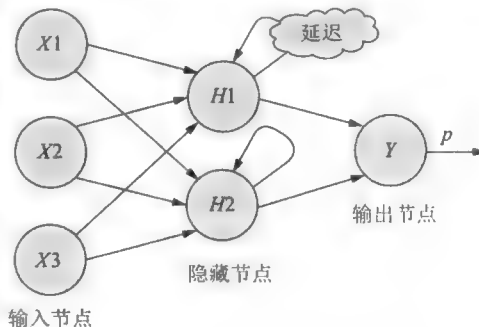
2. 信息传播的方向

你可能已经注意到，在前面的例子中，箭头用来指示信号只在一个方向上传播。如果网络中的输入信号在一个方向上从一个节点到另一个节点连续地传送，直到到达输出层，那么这样的网络称为**前馈网络**（feedforward network）。

虽然信息流有限制，但前馈网络提供了大量令人吃惊的灵活性。例如，层数和每一层的节点数都可以改变，多个结果可以同时进行建模，或者可以应用多个隐藏层（这种做法有时称为**深度学习**（deep learning））。



相比之下，**递归网络**（recurrent network）（或者**反馈网络**，feedback network）允许信号使用循环在两个方向上传播。这个性质更贴近地模拟了生物神经网络的工作原理，它使得极其复杂的模式可以被学习。增加一个短期记忆（下图中标记为**延迟**（Delay））会给递归网络增加巨大的功效。值得注意的是，这包含了能够理解经过一段时间的事件序列的能力。因此，递归网络可用于股市预测（stock market prediction）、语言理解（speech comprehension）和天气预报（weather forecasting）。一个简单递归网络的描述如右图所示。



尽管递归网络很有潜力，但是它们在很大程度上仍然是理论上的，在实践中很少使用。另一方面，前馈网络已经广泛地应用于现实世界的问题中。实际上，多层前馈网络（有时称为**多层感知器**（Multilayer Perceptron, MLP））是人工神经网络拓扑结构的事实标准。如果有人正在拟合一个神经网络而无需额外的说明，那么他们最有可能指的是多层前馈网络。

3. 每一层的节点数

除了层数和信息传播方向的变化外，神经网络同样可以改变每一层的节点数，从而导致复杂性发生改变。输入节点的个数由输入数据特征的数量预先确定。类似地，输出节点的个数由需要进行建模的结果或者结果中的分类水平数预先确定。然而，隐藏节点的个数留给使用者在训练模型之前确定。

不幸的是，没有可信的规则来确定隐藏层中神经元的个数。合适的数目取决于输入节点的个数、训练数据的数量、噪声数据的数量，以及许多其他因素之间的学习任务的复杂性。

一般情况下，更复杂的网络拓扑结构具有更多数目的网络连接，允许更复杂问题的学习。较多数量的神经元将产生反映训练数据更严格的模型，但有过度拟合的风险，而且它可能不能充分地推广到未来的数据。此外，大型神经网络计算量也很大，而且训练缓慢。

最好的做法就是基于验证数据集，使用较少的节点产生适用（足够）的性能。在大多数情况下，即使只有少量的隐藏节点（往往少到屈指可数），但神经网络却可以提供惊人（巨大）的学习能力。



已经证明，具有至少一个充分多神经元隐藏层的神经网络是一种**通用函数逼近器**（universal function approximator）。从本质上讲，这意味着这样的一个网络可以用来以任意精度逼近有界区间上的任意连续函数。

7.1.4 用后向传播训练神经网络

网络拓扑结构是一块空白石板，通过它本身并没有学到任何东西。就像一个刚出生的孩子，它必须用经验进行训练。当神经网络处理输入数据时，神经元之间的连接被加强或者减弱，类似于一个婴儿在体验外界环境时，他大脑的发育过程。网络的连接权重反映了观察到的随时间变化的模式。

通过调整连接权重训练神经网络模型的计算量非常大。因此，尽管人工神经网络之前已经被研究了几十年，但是很少将它们应用到真实世界的学习任务中，直到 20 世纪 80 年代中后期，一种有效的训练人工神经网络的方法被发现。该算法使用了一种后向传播误差的策略，简称为**后向传播**（backpropagation）。



有趣的是，同时期的几个研究团队相互独立地发现了后向传播算法。关于后向传播的开创性论文无疑是：Learning representations by back-propagating errors, Nature Vol. 323, pp. 533-566, by D.E. Rumelhart, G.E. Hinton, and R.J. Williams (1986)。

虽然相对于许多其他的机器学习算法，后向传播算法还是出了名的慢，但是该方法使得人们对于人工神经网络的兴趣再度升起。所以，现在使用后向传播算法的多层前馈网络在数据挖掘领域是常见的。这类模型具有如下的优点和缺点。

优点	缺点
<ul style="list-style-type: none"> ● 适用于分类和数值预测问题 ● 属于最精确的建模方法 ● 对数据的基本关系几乎不需要做出假设 	<ul style="list-style-type: none"> ● 计算量大，训练缓慢，特别是在网络拓扑结构复杂的情况下 ● 很容易过度拟合或者不充分拟合训练数据 ● 如果不是不可能，复杂黑箱模型的结果很难解释

在其最一般的形式中，后向传播算法通过两个过程的多次循环进行迭代。该算法的每一次迭代称为一个**新纪元**（epoch）。因为网络不包含先验的（a priori）（已有的）知识，所以通常在开始之前随机设定权重。然后，算法通过过程循环，直到达到一个停止准则。该循环包括：

- 在**前向阶段**（forward phase）中，神经元在从输入层到输出层的序列中被激活，沿途应用每一个神经元的权重和激活函数，一旦到达最后一层，就产生一个输出信号。
- 在**后向阶段**（backward phase）中，由前向阶段产生的网络输出信号与训练数据中的真实目标值进行比较，网络的输出信号与真实目标值之间的差异产生的误差在网络中向后传播，从而来修正神经元之间的连接权重，并减少将来产生的误差。

随着时间的推移，网络使用向后发送的信息来减少网络的总误差。然而，还有一个问题：因为每个神经元的输入和输出之间的关系很复杂，所以该算法如何确定一个权重需要改变多少（或者是否需要改变）呢？

回答这个问题涉及一个称为**梯度下降法**（gradient descent）的技术。从概念上讲，它的运作方式类似于一个被困于丛林中的探险者如何找到一条通向水源的路线——通过研究地形，在具有最大向下斜坡的方向上不断地走，他很有可能最终到达最低谷，而这很可能是一条河床。

在一个类似的过程中，后向传播算法利用每一个神经元的激活函数的导数来确定每一个输入权重方向上的梯度——因此，有一个可微的激活函数很重要，梯度将会因为权重的改变而表明误差是如何急剧减少或者增加的。该算法将试图通过一个称为**学习率**（learning rate）的量来改变权重以使得误差最大化地减小。学习率越大，算法试图降下的梯度就越快，这可以减少训练冒着风险越过山谷的时间。

虽然这个过程看起来很复杂，但在实践中很容易应用。让我们把对于多层前馈网络的理解应用到现实世界的问题中。

7.2 用人工神经网络对混凝土的强度进行建模

在工程领域中，对建筑材料的性能有精确的估计至关重要。这些估计是必需的，以便制定安全准则来管理用于楼宇、桥梁和道路建设中的材料。

估计混凝土的强度是一个特别有趣的挑战。尽管混凝土几乎要用于每一个建设项目，但由于它各种成分的使用以复杂的方式相互作用，所以它的性能变化很大。因此，很难精确地预测它最终产品的强度。给定一份输入材料成分清单，能够可靠地预测混凝土强度的模型可以带来更安全的建设行为。

7.2.1 第1步——收集数据

为了便于分析，使用由 I-Cheng Yeh 捐赠给 UCI 机器学习数据仓库（UCI Machine

Learning Data Repository) (<http://archive.ics.uci.edu/ml>) 的关于混凝土抗压强度的数据。因为 I-Cheng Yeh 发现用神经网络对这些数据进行建模是成功的, 所以我们将尝试使用 R 中的一个简单的神经网络模型来重复他的工作。



关于 I-Cheng Yeh 处理学习任务的更多信息, 请参考: Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research, Vol. 28, pp. 1797-1808, by I-C Yeh (1998).

根据该网站, 混凝土数据集包含了 1030 个混凝土案例, 8 个描述混合物成分的特征。这些特征被认为与最终的抗压强度相关, 并且它们包含了产品中使用的水泥 (cement)、矿渣 (slag)、灰 (ash)、水 (water)、超塑化剂 (superplasticizer)、粗集料 (coarse aggregate) 和细集料 (fine aggregate) 的量 (单位为 kg/m^3), 还包括老化时间 (aging time) (单位为天)。



要理解这个例子, 你需要从 Packt 出版社的网站上下载 concrete.csv 文件, 并将该文件保存到 R 的工作目录中

7.2.2 第 2 步——探索和准备数据

通常, 我们将通过使用 `read.csv()` 函数将数据加载为一个 R 对象, 并确认其符合预期的结构后, 开始我们的分析:

```
> concrete <- read.csv("concrete.csv")
> str(concrete)
'data.frame':    1030 obs. of  9 variables:
 $ cement      : num  141 169 250 266 155 ...
 $ slag        : num  212 42.2 0 114 183.4 ...
 $ ash          : num  0 124.3 95.7 0 0 ...
 $ water        : num  204 158 187 228 193 ...
 $ superplastic: num  0 10.8 5.5 0 9.1 0 0 6.4 0 9 ...
 $ coarseagg    : num  972 1081 957 932 1047 ...
 $ fineagg      : num  748 796 861 670 697 ...
 $ age          : int  28 14 28 28 28 90 7 56 28 28 ...
 $ strength     : num  29.9 23.5 29.2 45.9 18.3 ...
```

数据框中的 9 个变量对应于数据集中的 8 个特征和 1 个结果, 虽然已经产生一个很明显的问题。神经网络的运行最好是输入数据缩放到 0 附近的狭窄范围内, 但是这里我们所看到的数值范围大概是从 0 到 1000 多。

通常, 解决这个问题的方法是用规范化或者标准化函数来重新调整数据。如果数据服从一个钟形曲线 (如第 2 章描述的正态分布), 那么使用 R 内置的 `scale()` 函数才可能是有意义的。另一方面, 如果数据服从均匀分布或者严重非正态, 那么将其标准化到一个 0 ~ 1 范围可能会更合适。在这种情况下, 我们将使用后者。

在第3章中，我们自定义的 `normalize()` 函数为：

```
> normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

执行此代码后，使用 `lapply()` 函数，我们的 `normalize()` 函数就可以应用于混凝土数据框的每一列，如下所示：

```
> concrete_norm <- as.data.frame(lapply(concrete, normalize))
```

为了确认标准化确实运行了，我们看到最小强度和最大强度现在分别为 0 和 1：

```
> summary(concrete_norm$strength)
      Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000000 0.2663511 0.4000872 0.4171915 0.5457207 1.0000000
```

作为对比，原始的最小值和最大值分别为 2.33 和 82.6：

```
> summary(concrete$strength)
      Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
2.330000 23.71000 34.44500 35.81796 46.13500 82.60000
```



训练模型之前应用于数据的任何变换，之后需要应用反变换，以便将数据转换回原始的测量单位。为了便于重新调整，保存原始数据或者至少保存原始数据的主要统计量是明智的。

按照 I-Cheng Yeh 原始发表的范例，将数据划分为一个具有 75% 案例的训练集和一个具有 25% 案例的测试集。所使用的 CSV 文件已经以随机的顺序排列，所以我们只需要将其分成两部分：

```
> concrete_train <- concrete_norm[1:773, ]
> concrete_test <- concrete_norm[774:1030, ]
```

我们将使用训练数据集来创建神经网络，使用测试数据集来评估模型推广到未来的结果有多好。因为很容易过度拟合神经网络，所以这个步骤非常重要。

7.2.3 第3步——基于数据训练模型

为了对混凝土中使用的原料和最终产品的强度之间的关系建立模型，我们将使用一个多层前馈神经网络。由 Stefan Fritsch 和 Frauke Guenther 创建的 `neuralnet` 添加包提供了一个标准的易于使用的网络实现，而且该添加包还提供了一个函数用来绘制网络拓扑结构。由于这些原因，`neuralnet` 添加包的实现对于学习更多关于神经网络的知识是一个强大的选择。不过，这并不是说不能用它来很好地完成实际工作——很快你就会看到，这是一个相当强大的工具。

因为 `neuralnet` 添加包没有包含在基本 R 中，所以需要通过输入命令 `install.packages("neuralnet")` 来安装，并使用 `library(neuralnet)` 命令将其加载到 R

中。使用下面文本框中的语法，所包含的 `neuralnet()` 函数就可以用来训练用于数值预测的神经网络。



R 中还有一些其他常用的添加包来训练人工神经网络模型，每个添加包都有其独特的优势和劣势。因为 `nnet` 添加包的安装是作为标准 R 安装的一部分，所以它可能是最经常用来实现人工神经网络的添加包。它使用一种比标准的后向传播算法略微复杂的算法。另一个强大的选择是 `RSNNS` 添加包，它提供了一套完整的神经网络功能，其不利的一面就是学习起来更难。

神经网络语法

应用 `neuralnet` 添加包中的 `neuralnet()` 函数

建立模型：

```
m <- neuralnet(target ~ predictors, data = mydata, hidden = 1)
```

- `target`: 是数据框 `mydata` 中需要建模的输出变量
- `predictors`: 是给出数据框 `mydata` 中用于预测的特征的一个 R 公式
- `data`: 给出包含变量 `target` 和 `predictors` 的数据框
- `hidden`: 给出隐藏层中神经元的数目（默认为 1）

进行预测：

```
p <- compute(m, test)
```

- `m`: 函数 `neuralnet()` 所训练的模型
- `test`: 包含测试数据的数据框，它具有和用于训练模型的训练数据相同的特征

该函数返回一个两元素的列表：`$neurons`，用于保存神经网络每一层的神经元；`$net.result`，用于保存模型的预测值。

例子：

```
concrete_model <- neuralnet(strength ~ cement + slag + ash, data = concrete)
model_results <- compute(concrete_model, concrete_data)
strength_predictions <- model_results$net.result
```

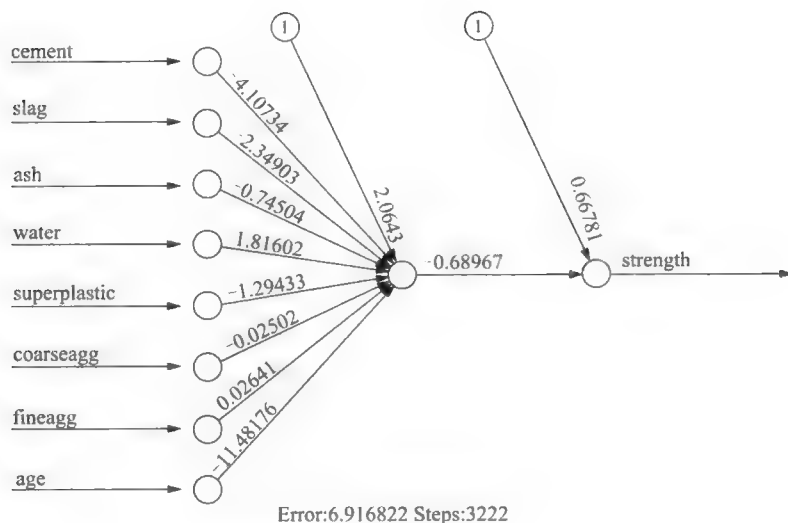
我们将从训练最简单的只有一个单一隐藏节点的多层前馈网络开始：

```
> concrete_model <- neuralnet(strength ~ cement + slag +
  ash + water + superplastic +
  coarseagg + fineagg + age,
  data = concrete_train)
```

然后，基于 `concrete_model` 对象使用 `plot()` 函数将网络拓扑结构可视化：

```
> plot(concrete_model)
```

在这个简单的模型中，对于 8 个特征中的每一个特征都有一个输入节点，后面跟着一个单一的隐藏节点和一个单一的预测混凝土强度的输出节点。每一个连接的权重也都被描绘出来，偏差项也被描绘出来（通过带有 1 的节点表示）。该图还报告了训练的步数和一个称为误差平方和（Sum of Squared Error, SSE）的度量指标。当评估模型的性能时，这些指标很有用。



7.2.4 第4步——评估模型的性能

网络拓扑结构图让我们窥视了人工神经网络的黑箱，但是它并没有提供更多关于模型拟合数据好坏的信息。为了评估模型的性能，可以基于测试数据集使用 `compute()` 函数生成预测：

```
> model_results <- compute(concrete_model, concrete_test[1:8])
```

注意，`compute()` 函数的运行原理与我们已经使用至今的 `predict()` 函数有些不同。它返回一个带有两个分量的列表：`$neurons`，用来存储网络中每一层的神经元；`$net.results`，用来存储预测值。我们想要的是后者：

```
> predicted_strength <- model_results$net.result
```

因为这是数值预测问题而不是分类问题，所以不能用混淆矩阵来检查模型的准确性。相反，我们必须度量我们预测的混凝土强度与其真实值之间的相关性，这可以让我们深入了解这两个变量之间线性相关的强度。

回想一下，`cor()` 函数可用来获取两个数值向量之间的相关性：

```
> cor(predicted_strength, concrete_test$strength)
[1,]
[1,] 0.7170368646
```



如果你的结果不同，请不要惊慌。因为神经网络开始于随机的权重，所以模型之间的预测值可以有所不同。

相关性接近 1 表示两个变量之间具有很强的线性关系。因此，这里大约为 0.72 的相关性表示具有一个相当强的线性关系。这意味着即使只有一个单一的隐藏节点，我们的模型做了

相当不错的工作



具有一个隐藏节点的神经网络与第6章学习的线性回归模型类似，每个输入节点与隐藏节点之间的权重类似于回归系数，对偏差项的权重类似于截距。事实上，如果构建一个与前面的神经网络原理相同的线性模型，则相关系数为 0.74。

考虑到只使用了一个隐藏节点，因此很有可能提高我们模型的性能。让我们试着建立一个更好的模型。

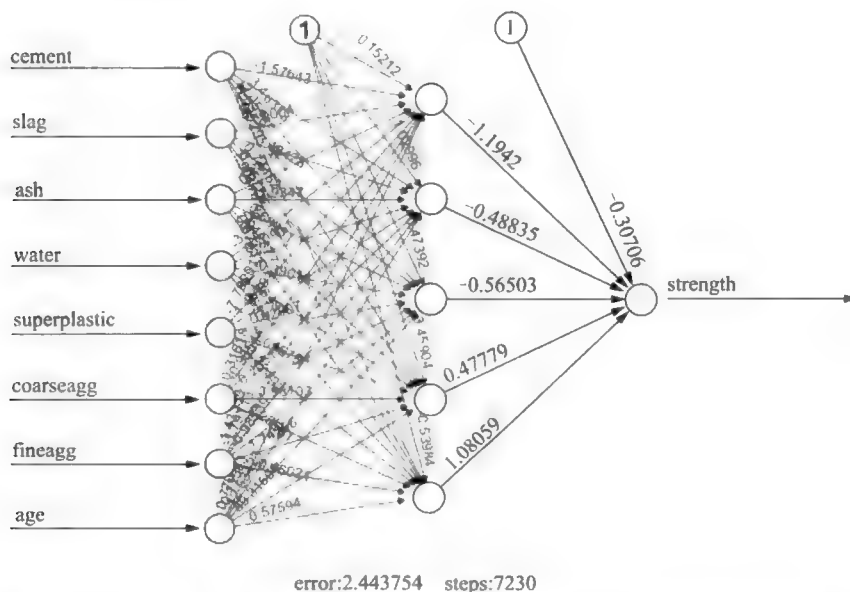
7.2.5 第5步——提高模型的性能

因为具有更复杂拓扑结构的网络能够学习更艰难的概念，所以我们看看，当隐藏节点的个数增加到 5 时，会发生什么。与之前一样使用 `neuralnet()` 函数，但增加参数 `hidden = 5`：

```
> concrete_model2 <- neuralnet(strength ~ cement + slag +
  ash + water + superplastic +
  coarseagg + fineagg + age,
  data = concrete_train, hidden = 5)
```

再次绘制网络图，可以看到连接的数量急剧增加。这如何影响模型的性能呢？

```
> plot(concrete_model2)
```



注意，所报告的误差（依然是通过 SSE 度量）已经从之前模型的 6.92 减少为这里的 2.44。此外，训练的步数从 3222 步上升为 7230 步，考虑到现在的模型已经变得多复杂后，

这也就不足为奇了。

采用相同的步骤对预测值和真实值进行比较，现在我们获取的相关系数大约为 0.8，与之前的结果相比，这是一个相当大的改进。

```
> model_results2 <- compute(concrete_model2, concrete_test[1:8])
> predicted_strength2 <- model_results2$net.result
> cor(predicted_strength2, concrete_test$strength)
      [,1]
[1,] 0.801444583
```

有趣的是，在原始的文章中，I-Cheng Yeh 使用了一个非常相似的神经网络，其报告的平均相关性为 0.855。由于某种原因，我们得到的相关系数稍差一点儿。我们的观点是，I-Cheng Yeh 是土木工程教授，因此他可能已经在数据准备中应用了一些学科专业知识。如果你想用神经网络做更多的实践，你可以尝试应用本章前面学到的原则，或者可以通过使用不同数量的隐藏节点，或者采用不同的激活函数等，来战胜他的结果。此外，`?neuralnet` 帮助页面提供了更多关于可调整的各种参数的信息。

7.3 理解支持向量机

支持向量机 (Support Vector Machine, SVM) 可以想象成一个平面，该平面定义了各个数据点之间的界线，而这些数据点代表根据它们的特征值绘制在多维空间中的案例。支持向量机的目标是创建一个平面边界，称为一个超平面 (hyperplane)，使得任何一边的数据划分都是相当均匀的。通过这种方式，支持向量机学习结合了第 3 章中呈现的基于实例的近邻学习和第 6 章中描述的线性回归建模两个方面，这种结合是极其强大的，允许支持向量机对非常复杂的关系进行建模。

虽然推动支持向量机的数学基础已经存在了几十年，但是它们近来才人气暴涨。这当然植根于它们最先进的性能，但或许也是因为这个事实，即屡获殊荣的支持向量机算法已经通过许多程序语言（包括 R）在一些受到欢迎和大力支持的库中得到实现。这使得支持向量机得到更广泛的用户接受，而这些用户之前可能因为支持向量机的实现涉及比较复杂的数学而不去用它。然而，好消息是，尽管数学可能很难，但是基本的概念是可以理解的。

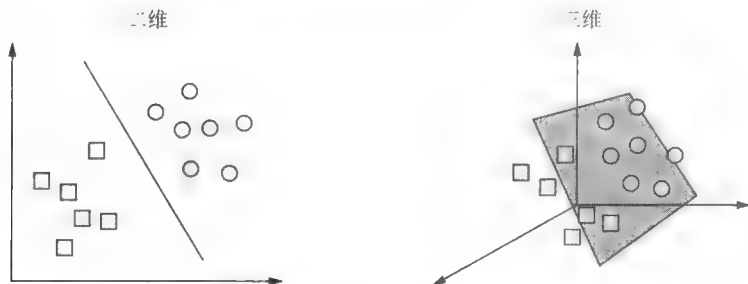
支持向量机几乎可以适用于所有的学习任务，包括分类和数值预测两个方面。许多算法成功的关键都是来自模式识别。著名的应用包括：

- ❑ 在生物信息学领域中，识别癌症或者其他遗传疾病的微阵列基因表达数据的分类。
- ❑ 文本分类，比如根据主题鉴定用于某个文档或者组织文档的语言。
- ❑ 罕见却重要的事件检测，如内燃机故障、安全漏洞或者地震等。

当支持向量机用于二元分类时，它最容易理解，这就是为什么该方法已经被习惯应用的原因。因此，在剩下的部分，我们将只专注于支持向量机分类器。然而，不用担心，当支持向量机适用于其他学习任务（比如，数值预测）时，这里学习的原则同样适用。

7.3.1 用超平面分类

正如前面所指出的，支持向量机使用一种称为超平面的线性边界将数据划分成具有相似元素的组，通常由类值表示。例如，下图描绘了一个超平面在二维和三维空间中将数据分成了圆形组和正方形组。由于圆形和正方形可以由一条直线或者一个平面进行划分，所以它们是**线性可分的**（linearly separable）。起初，我们只考虑简单的情况。在简单的情况下，这是正确的，但支持向量机同样可以扩展到数据不是线性可分的问题。



为方便起见，在二维空间中，超平面通常被描绘成一条线，但这仅仅是因为在大于二维的空间中，这很难说明。在现实中，超平面在高维空间中是一个平面——一个可能很难让你理解的概念。

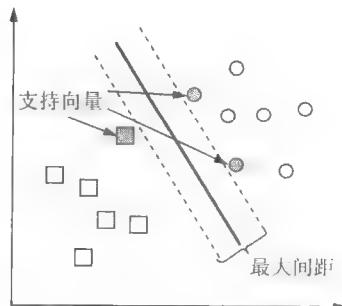
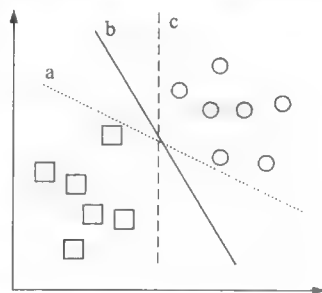
支持向量机算法的任务就是确定一条用于分隔两个类别的线。如下图所示，圆形组和正方形组之间的分隔线不止一种选择，有3种标记为a、b和c的选择。那么支持向量机算法该怎样选择呢？

7.3.2 寻找最大间隔

回答上面的问题涉及寻找创建两个类之间最大间隔的**最大间隔超平面**（Maximum Margin Hyperplane, MMH）。尽管分隔圆形和正方形的3条线中的任意一条都将对所有的数据点进行正确分类，但很有可能产生最大间隔的那条线将能够最好地推广到将来的数据。这是因为在边界附近点位置的微小变化可能会导致这些点中的某些点碰巧落在线之外。

支持向量（support vector）（由下图中的箭头所示）是每个类中最接近最大间隔超平面的点，而且每类必须至少有一个支持向量，但也可能有多个。单独使用支持向量，就可以定义最大间隔超平面，这是支持向量机的一个重要特征。支持向量机提供了一种非常简洁（紧凑）的方式来存储分类模型，即使特征的个数非常多。

用来确定支持向量的算法依赖于向量几何，并涉及本书



范围之外的一些相当棘手（复杂）的数学问题。然而，这个过程的基本原理是相当简单的。

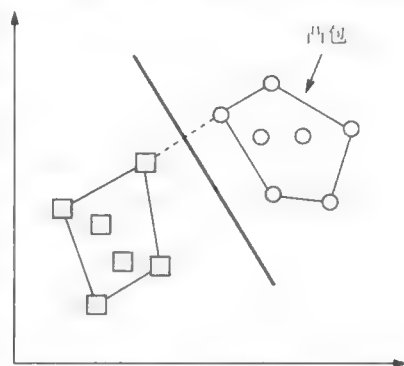


更多关于支持向量机的数学论述可以在下面这篇经典论文中找到：Support-vector network, Machine Learning, Vol. 20, pp. 273-297, by C. Cortes and V. Vapnik (1995) 初学者水平的探讨可参见：Support vector machines: hype or hallelujah, SIGKDD Explorations, Vol. 2, No. 2, pp. 1-13, by K.P. Bennett and C. Campbell (2003) 更深入的了解，可参见：Support Vector Machines by I. Steinwart and A. Christmann (Springer Publishing Company, 2008).

1. 线性可分的数据情况

在类是线性可分的假设下，如何找到最大间隔是最容易理解的。在这种情况下，最大间隔超平面要尽可能地远离两组数据点的外边界，这些外边界称为凸包（convex hull）。然后，最大间隔超平面就是两个凸包之间最短距离直线的垂直平分线。使用一种称为二次优化（quadratic optimization）的技术，复杂的计算机算法就能够通过这种方式找到最大间隔。

另一种等价的替代方法涉及通过每一个可能的超平面的空间搜索，从而找到一组将数据划分成同类组的两个平行平面，但这两个平面本身却要尽可能地远离。换言之，这个过程有点儿像试图找到能够适合从楼梯井上搬到卧室的最大床垫。



要理解这个搜索过程，需要通过一个超平面来确切定义上面所叙述的过程。在 n 维空间中，使用下面的方程：

$$\vec{w} \cdot \vec{x} + b = 0$$

如果你对这种表示法不熟悉，那么你需要知道字母上方的箭头表明它们是向量而不是数字。特别地， \vec{w} 是一个 n 维的权重向量，即 $\{w_1, w_2, \dots, w_n\}$ ，而 b 是一个称为偏差的单一的数字。



如果你感到困惑或者难以想象平面，那么不必担心这些细节，可以简单地认为这个方程是定义一条线的一种方式，与斜截式（ $y = mx + b$ ）用于定义 2 维空间中的直线一样。

利用这个公式，该过程的目标就是找到一组指定两个超平面的权重，如下所示：

$$\begin{aligned}\vec{w} \cdot \vec{x} + b &\geq +1 \\ \vec{w} \cdot \vec{x} + b &\leq -1\end{aligned}$$

同样，我们将要求这两个指定的超平面使得第一类中所有的点落在第一个超平面的上方，另一类中所有的点落在第二个超平面的下方。只要数据是可分的，这样做就是可能的。

向量几何定义这两个平面之间的距离为：

$$\frac{2}{\|\vec{w}\|}$$

这里， $\|\vec{w}\|$ 表示欧几里得范数 (Euclidean norm) (从原点到向量 \vec{w} 的距离)，因此，要最大化距离，我们需要最小化 $\|\vec{w}\|$ 。为了便于找到解决方案，这类任务通常重新表述为一组约束：

$$\min \frac{1}{2} \|\vec{w}\|^2$$

$$\text{使得 } y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall \vec{x}_i$$

尽管这看起来难以处理，但是如果你分块思考，就不会太复杂。基本上，这种想法就是在使得每一个数据点 y_i 正确分类的条件下，最小化前面的公式。注意， y 表示分类值 (转换为 +1, 或 -1)，此外，倒置的“A”表示“所有”(for all) 的

与用其他方法寻找最大间隔一样，寻找该问题的解决方案是二次优化软件的一项工作。尽管它能够处理密集型的数据，但是专门的算法即使对于相当大的数据集，也能够很快解决这些问题。

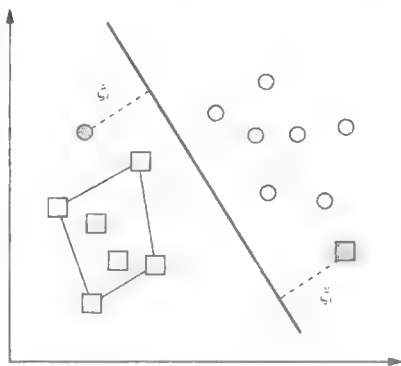
2. 非线性可分的数据情况

当研究过支持向量机背后的理论后，你可能想知道关于房间里的大象：在数据不是线性可分的情况下，会发生什么？这个问题的解决方案使用了一个松弛变量 (slack variable)，这样就创建了一个软间隔，允许一些点落在线不正确的一边。下图显示两个点落在了与松弛项 (用希腊字母 ξ_i 表示) 相对应的线的错误边：

用成本值 (记为 C) 表示所有违反约束的点，而且该算法试图使总成本最小，而不是寻找最大间隔。因此，可以修正优化问题：

$$\min \frac{1}{2} \|\vec{w}\|^2 + C \sum_{i=1}^n \xi_i$$

$$\text{使得 } y_i (\vec{w} \cdot \vec{x}_i - b) \geq 1 - \xi_i \quad \forall \vec{x}_i, \xi_i \geq 0$$

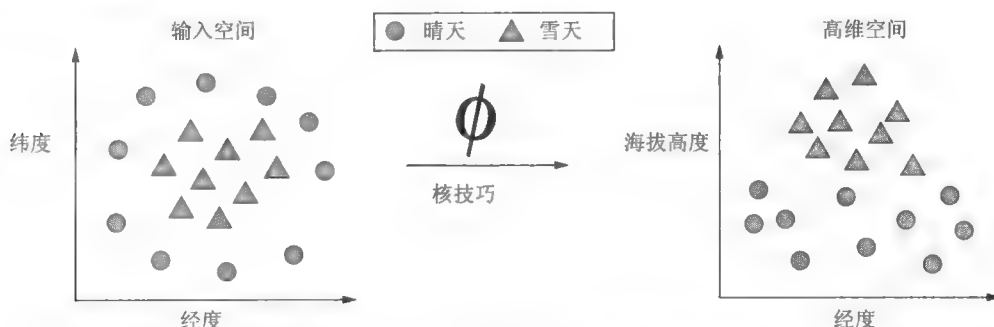


如果你现在感到困惑，请不用担心，你并不孤单 (很多人都感到困惑)。幸运的是，支持向量机软件包将会愉快地为你解决优化问题，而你无需理解技术细节。需要理解的重要一点是增加的成本参数 C ，修改这个值将调整对于落在超平面错误一边的案例的惩罚。成本参数越大，努力实现 100% 分离的优化就会越困难。另一方面，较小的成本参数将把重点放在更宽 (广泛) 的整体边缘。为了创建可以很好概括未来数据的模型，在这两者之间取得平衡是非常重要的。

7.3.3 对非线性空间使用核函数

在许多现实世界的应用中，变量之间的关系是非线性的。正如我们刚刚所发现的，基于这样的数据，支持向量机通过添加一个松弛变量后仍然可以被训练，这允许一些案例被错误地分类。然而，这并不是处理非线性问题的唯一方式。支持向量机的一个关键特征就是它们能够使用一种称为**核技巧**（kernel trick）的处理方式将问题映射到一个更高维的空间中。如果这样做，非线性关系可能会突然看起来是完全线性的。

虽然这似乎是没有意义的，但通过示例，其实很容易说明。在下面的图中，左边的图描绘了一个天气类（晴天或者雪天）与两个特征（**经度**（Latitude）和**纬度**（Longitude））之间的非线性关系。该图中心的点是**雪天**（Snowy）类的成员，而边缘的点全都是**晴天**（Sunny）。这样的数据可能产生于一组天气报告，其中一些来源于山顶附近的基地，而另一些则来源于山脚周围的基地。



右边的图，在使用了核技巧后，可以通过一个新维度（**海拔高度**，Altitude）的视角看数据。通过添加这个特征，现在这些类完全线性可分。这之所以可能，是因为我们已经获得了一个看数据的新视角。在左图中，我们是从一只鸟的视角看山，而在右图中，我们是从地平面看山。这里，趋势很明显了：在更高的海拔高度就发现了雪天。

具有非线性核的支持向量机通过对数据添加额外的维度，以便以这种方式创建分离。从本质上讲，核技巧涉及一个添加能够表述度量特征之间数学关系新特征的过程。例如，高度特征在数学上可以表示为纬度和经度之间的一个相互作用——点越接近于这些尺度的每一个的中心，高度就越高，这使得支持向量机可以学习原始数据中未明确度量的概念。

具有非线性核的支持向量机是极其强大的分类器，虽然它们也确实有一些缺点。其优缺点如下表所示：

优点	缺点
<ul style="list-style-type: none"> ● 可用于分类或者数值预测问题 ● 不会过多地受到噪声数据的影响，而且不容易出现过度拟合 ● 可能比神经网络更容易使用，特别是由于几个得到很好支持的支持向量机算法的存在 ● 由于它的准确度高，而且在数据挖掘竞赛中高调地胜利，所以越来越受欢迎 	<ul style="list-style-type: none"> ● 寻找最好的模型需要测试不同的核函数和模型参数的组合 ● 训练缓慢，尤其是输入数据集具有大量的特征或者案例时 ● 导致一个复杂的黑箱模型，很难，甚至无法解释

在一般情况下，核函数是下面的形式。这里，该函数用希腊字母 phi 表示，即 $\phi(x)$ ，是一个将数据转换到另一个空间的映射：

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

使用这种形式，核函数已经用于许多不同领域的的数据。最常用的几个核函数列举如下。几乎所有的支持向量机软件包都将包括这些核，以及许多其他的核。

线性核函数 (linear kernel) 根本不需要转换数据，因此它可以简单地表示为特征的点积：

$$K(\vec{x}_i, \vec{x}_j) = \vec{x}_i \cdot \vec{x}_j$$

次数为 d 的**多项式核函数** (polynomial kernel) 添加了一个简单的非线性数据变换：

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^d$$

S 形核函数 (sigmoid kernel) 产生支持向量机模型，类似于神经网络使用 S 形激活函数。希腊字母 kappa 和 delta 用来作为核参数：

$$K(\vec{x}_i, \vec{x}_j) = \tanh(k \vec{x}_i \cdot \vec{x}_j - \delta)$$

高斯 RBF 核函数 (Gaussian RBF kernel) 类似于 RBF 神经网络。RBF 核函数对于许多类型的数据都运行得很好，而且被认为是用于许多学习任务的一个合理的开始：

$$K(\vec{x}_i, \vec{x}_j) = e^{-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}}$$

对于特定的任务，没有可以依赖的规则用于匹配核函数。在很大程度上拟合取决于要学习的概念以及训练数据的量和特征之间的关系。通常情况下，一点点的试验和误差需要基于验证数据集训练和评估多个支持向量机。也就是说，在许多情况下，核函数的选择是任意的，因为性能可能只有轻微的变化。为了看到在实际中它是如何运行的，下面将我们对支持向量机分类的理解应用于一个现实世界的问题。

7.4 用支持向量机进行光学字符识别

对许多类型的机器学习算法来说，图像处理都是一项艰巨的任务。将像素模式连接到更高概念的关系是极其复杂的，而且很难定义。例如，让一个人识别一张面孔、一只猫或者字母 A 是很容易的，但用严格的规则来定义这些模式是很困难的。此外，图像数据往往是噪声数据。关于如何捕获图像，有许多细微的变化，这取决于灯光、定位和对象的位置。

支持向量机非常适合处理图像数据带来的挑战，它们能够学习复杂的图案而不需要对噪声过度敏感，它们能够以高的准确度识别光学图案。而且，支持向量机的主要缺点（黑箱模型的代表），对于图像处理并不那么重要。如果一个支持向量机能够区分一只猫和一只狗，那么它是如何做到的并不很重要。

在本节中，我们将研究一个模型，该模型类似于那些往往与桌面文档扫描仪捆绑在一起的光学字符识别（Optical Character Recognition, OCR）软件的核心模型。此类软件的目的是通过将印刷或者手写文本转换成一种电子形式，保存在数据库中来处理纸质文件。当然，由于手写风格和印刷字体有许多变体，所以这是困难的问题。即便如此，软件用户还是期待完美，因为纸漏或者拼写错误可能会导致商业环境中的尴尬或者代价高昂的过失（错误）。让我们来看看支持向量机是否能够胜任这项任务。

7.4.1 第1步——收集数据

当光学字符识别软件第一次处理文件时，它将文件划分成一个矩阵，从而网格中的每一个单元包含一个单一的图像字符（glyph），这是一种适用于字母、符号或者数字的好方式。接着，对于每一个单元，该软件将试图对一组它能识别的所有字符进行图像字符匹配。最后，单个字符将重新在一起组合成词，这可以用文档语言中的字典来有选择地进行拼写检查。

在这个练习中，假设我们已经开发了将文件分割成矩形区域，每一个区域包含一个单一字符的算法；还假设文件中只包含英文字母字符。因此，我们将模拟一个过程，涉及对从 A ~ Z 的 26 个字母中的一个进行图像字符匹配。

为此，使用由 W. Frey 和 D. J. Slate 捐赠给 UCI 机器学习数据仓库（UCI Machine Learning Data Repository）（<http://archive.ics.uci.edu/ml>）的一个数据集。该数据集包含了 26 个大写英文字母的 2000 个案例，使用 20 种不同的随机重塑和扭曲的黑色和白色字体印刷。



有关这些数据的更多信息，请参考：Letter recognition using Holland-style adaptive classifiers, Machine Learning, Vol. 6, pp. 161-182, by W. Frey and D.J. Slate (1991).

下图由 W. Frey 和 D. J. Slate 发布，提供了一个包含一些印刷图像字符的案例。这种方式的扭曲，用计算机识别字母是具有挑战性的，但这些字母却很容易被人识别：



7.4.2 第2步——探索和准备数据

根据 Frey 和 Slate 提供的文件，当图像字符被扫描到计算机中，它们将转换成像素，并且有 16 个统计属性：

这些属性用图像字符的水平和垂直尺寸、黑色（相对于白色）像素的比例、像素的平均水平与垂直位置来测量字符。据推测，字符所构成的盒子的不同区域的黑色像素浓度的差异应该提供了一种区分字母表中 26 个字母的方法。



要理解这个例子，你需要从 Packt 出版社的网站下载 letterdata.csv 文件，并将该文件保存到 R 的工作目录中。

将数据读到 R 中，确认接收到的数据具有 16 个特征，这些特征定义了每一个字母类的

案例。正如预期的那样，letter 有 26 个水平：

```
> letters <- read.csv("letterdata.csv")
> str(letters)
'data.frame': 20000 obs. of 17 variables:
 $ letter: Factor w/ 26 levels "A","B","C","D",...
 $ xbox  : int  2 5 4 7 2 4 4 1 2 11 ...
 $ ybox  : int  8 12 11 11 1 11 2 1 2 15 ...
 $ width : int  3 3 6 6 3 5 5 3 4 13 ...
 $ height: int  5 7 8 6 1 8 4 2 4 9 ...
 $ onpix : int  1 2 6 3 1 3 4 1 2 7 ...
 $ xbar  : int  8 10 10 5 8 8 8 8 10 13 ...
 $ ybar  : int  13 5 6 9 6 8 7 2 6 2 ...
 $ x2bar : int  0 5 2 4 6 6 6 2 2 6 ...
 $ y2bar : int  6 4 6 6 6 9 6 2 6 2 ...
 $ xybar : int  6 13 10 4 6 5 7 8 12 12 ...
 $ x2ybar: int  10 3 3 4 5 6 6 2 4 1 ...
 $ xy2bar: int  8 9 7 10 9 6 6 8 8 9 ...
 $ xedge : int  0 2 3 6 1 0 2 1 1 8 ...
 $ xedgey: int  8 8 7 10 7 8 8 6 6 1 ...
 $ yedge : int  0 4 3 2 5 9 7 2 1 1 ...
 $ yedgex: int  8 10 9 8 10 7 10 7 7 8 ...
```

回想一下，支持向量机学习算法需要所有的特征都是数值型的，而且每一个特征需要缩小到一个相当小的区间中。在这种情况下，每一个特征都是一个整数，所以不需要将任意一个因子转换成数字。另一方面，这些整型变量的一些范围显现得相当宽，这似乎暗示需要标准化或者规范化数据。事实上，我们可以跳过这一步，因为用来拟合支持向量机模型的 R 添加包会自动帮助我们对数据进行重新调整。

考虑到大部分的数据准备都已经帮助我们完成，所以我们可以直接跳到机器学习过程的训练和测试阶段。在前面的分析中，需要在训练集和测试集之间随机地划分数据。尽管我们在这里可以做，但是 Frey 和 Slate 已经将数据随机化，并建议使用前 16 000 个记录（80%）来建立模型，使用后 4000 条记录（20%）来进行测试。按照他们的建议，我们可以创建训练数据框和测试数据框，如下代码所示：

```
> letters_train <- letters[1:16000, ]
> letters_test  <- letters[16001:20000, ]
```

既然数据准备好了，那就让我们开始建立分类器吧。

7.4.3 第 3 步——基于数据训练模型

当谈到在 R 中拟合支持向量机模型时，有几个突出的添加包可以选择。来自维也纳理工大学（Vienna University of Technology, TU Wien）统计系的 e1071 添加包提供了一个屡获殊荣的 LIBSVM 库的 R 接口，即一个用 C++ 编写的广泛使用的开源支持向量机程序。如果你已经熟悉 LIBSVM，你可能想从这里开始。



关于 LIBSVM 的更多信息，请参考该作者的网站：<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>。

同样,如果你已经投入到 SVMlight 算法中,那么来自多特蒙德工业大学(Dortmund University of Technology, TU Dortmund)统计系的 klaR 添加包提供了 R 中该支持向量机的函数实现。



关于 SVMlight 的信息,可以看一下: <http://svmlight.joachims.org/>。

最后,如果你是从头开始,那么用 kernlab 添加包中的支持向量机函数或许是最好的开始。这个添加包的一个有趣优点就是它原本就是在 R 中开发的,而不是在 C 或者 C++ 中开发的,这使得它可以很容易地设置,没有任何隐藏在幕后的内部结构。或许更重要的是,与其他的选择方案不同, kernlab 添加包可以与 caret 添加包一起使用,这就允许支持向量机模型可以使用各种自动化方法进行训练和评估(在第 11 章中有更深入的介绍)。



关于 kernlab 添加包更详尽的介绍,请参考如下网站中该作者的论文。

<http://www.jstatsoft.org/v11/i09/>。

用 kernlab 添加包来训练支持向量机分类器的语法如下所示。如果你碰巧使用其他添加包,那么这些命令在很大程度上也是相似的。默认情况下, ksvm() 函数使用高斯 RBF 核函数,但也提供了一些其他的选项。

支持向量机语法

应用 kernlab 添加包中的 ksvm() 函数

建立模型:

```
m <- ksvm(target ~ predictors, data = mydata, kernel = "rbfdot", c = 1)
```

- target: 是数据框 mydata 中需要建模的输出变量
- predictors: 是给出数据框 mydata 中用于预测的特征的一个 R 公式
- data: 给出包含变量 target 和 predictors 的数据框
- kernel: 给出隐一个非线性映射(mapping),例如 "rbfdot" (径向基函数), "polydot" (多项式函数), "tanhdot" (双曲正切函数), "vanilladot" (线性函数)
- c: 用于给出违反约束条件时的惩罚,即对于“软边界”的惩罚的大小。较大的 c 值将导致较窄的边界。该函数返回一个可以用于预测的 SVM 对象。

进行预测:

```
p <- predict(m, test, type = "response")
```

- m: 函数 ksvm() 所训练的模型
- test: 包含测试数据的数据框,它具有和用于训练模型的训练数据相同的特征
- type: 用于指定预测的类型为 "response" (预测类别),或者 "probabilities" (预测概率,每一列对应一个类水平值)

根据 type 参数的设定,该函数返回一个包含预测类别(或者概率)的向量(或者矩阵)、两元素的列表: \$neurons, 用于保存神经网络每一层的神经元; \$net.result, 用于保存模型的预测值。

例子:

```
letter_classifier <- ksvm(letter ~ ., data = letters_train, kernel = "vanilladot")
letter_prediction <- predict(letter_classifier, letters_test)
```

为了提供度量支持向量机性能的基准，我们从训练一个简单的线性支持向量机分类器开始。如果你还没有准备好，使用命令 `install.packages("kernlab")` 将 `kernlab` 添加包安装到系统中。然后，就可以基于训练数据调用 `ksvm()` 函数，并使用 `vanilladot` 选项指定线性核函数（即 `vanilla`），如下所示：

```
> library(kernlab)
> letter_classifier <- ksvm(letter ~ ., data = letters_train,
                           kernel = "vanilladot")
```

根据计算机的性能，这个运算可能需要一些时间来完成。当它完成后，输入存储模型的名称来看一看关于训练参数和模型拟合度的一些基本信息。

```
> letter_classifier
Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)
parameter : cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 7037

Objective Function Value : -14.1746 -20.0072 -23.5628 -6.2009 -7.5524
-32.7694 -49.9786 -18.1824 -62.1111 -32.7284 -16.2209...

Training error : 0.130062
```

这些信息几乎没有告诉我们关于模型在现实世界中运行好坏的程度。因此，我们需要根据测试数据集来研究模型的性能，从而判断它是否能够很好地推广到未知的数据。

7.4.4 第4步——评估模型的性能

`predict()` 函数允许我们基于测试数据集使用字母分类模型进行预测：

```
> letter_predictions <- predict(letter_classifier, letters_test)
```

因为我们没有指定 `type` 参数，所以默认使用了 `type = "response"`，这样就返回了一个向量。该向量包含对应于测试数据中每一行值的一个预测字母，使用 `head()` 函数，我们可以看到前 6 个预测字母是 U、N、V、X、N 和 H：

```
> head(letter_predictions)
[1] U N V X N H
Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

为了研究分类器的性能，我们需要将测试数据集中的预测值与真实值进行比较。为了这个目的，我们使用 `table()` 函数（这里只显示了全部表格的一部分）：

```
> table(letter_predictions, letters_test$letter)
letter_predictions   A   B   C   D   E
      A 144    0    0    0    0
      B   0 121    0    5    2
```

C	0	0	120	0	4
D	2	2	0	156	0
E	0	0	5	0	127

对角线的值 144、121、120、156 和 127 表示的是预测值与真实值相匹配的总记录数。同样，出错的数目也列出来了。例如，位于行 B 和列 D 的值 5 表示有 5 种情况将字母 D 误认为字母 B。

单个地看每个错误类型，可能会揭示一些有趣的关于模型识别有困难的特定字母类型的模式，但这也是很耗费时间的。因此，我们可以通过计算整体的准确度来简化我们的评估，即只考虑预测的字母是正确的还是不正确的，并忽略错误的类型。

下面的命令返回一个元素为 TRUE 或者 FALSE 值的向量，表示在测试数据集中，模型预测的字母是否与真实的字母相符（即匹配）。

```
> agreement <- letter_predictions == letters_test$letter
```

使用 table() 函数，我们看到，在 4000 个测试记录中，分类器正确识别的字母有 3357 个：

```
> table(agreement)
agreement
FALSE  TRUE
  643   3357
```

以百分比计算，准确度大约为 84%：

```
> prop.table(table(agreement))
agreement
FALSE    TRUE
0.16075 0.83925
```

注意，当 Frey 和 Slate 在 1991 年发布该数据集时，他们报告的识别准确度大约为 80%。仅仅使用了几行 R 代码，我们的结果便能够优于他们的结果，不过我们也受益于超过 20 年的额外的机器学习研究。考虑到这一点，我们很有可能可以做得更好。

7.4.5 第 5 步——提高模型的性能

之前的支持向量机模型使用简单的线性核函数。通过使用一个更复杂的核函数，我们可以将数据映射到一个更高维的空间，并有可能获得一个较好的模型拟合度。

然而，从许多不同的核函数进行选择是具有挑战性的。一个流行的惯例就是从高斯 RBF 核函数开始，因为它已经被证明对于许多类型的数据都能运行得很好。我们可以使用 ksvm() 函数来训练一个基于 RBF 的支持向量机，如下所示：

```
> letter_classifier_rbf <- ksvm(letter ~ ., data = letters_train,
                                kernel = "rbfdot")
```

然后，我们像之前一样进行预测：

```
> letter_predictions_rbf <- predict(letter_classifier_rbf,
                                     letters_test)
```

最后，与我们的线性支持向量机的准确度进行比较：

```
> agreement_rbf <- letter_predictions_rbf == letters_test$letter
> table(agreement_rbf)
agreement_rbf
FALSE   TRUE
  281   3719
> prop.table(table(agreement_rbf))
agreement_rbf
FALSE   TRUE
0.07025 0.92975
```

通过简单地改变核函数，我们可以将字符识别模型的准确度从 84% 提高到 93%。如果这种性能水平对于光学字符识别程序仍不能令人满意，那么你可以测试其他的核函数或者通过改变成本约束参数 C 来修正决策边界的宽度。作为练习，你应该用这些参数来不断尝试，看看它们如何影响最终的成功模型。

7.5 总结

在本章中，我们研究了两个能够提供巨大潜能的机器学习方法，但它们往往由于其复杂性而被忽视。但愿你现在意识到这样的名声至少是有点不当的，毕竟推动人工神经网络和支持向量机的基本概念是相当容易理解的。

另一方面，因为人工神经网络和支持向量机已经存在了几十年，所以它们中的每一个都有许多变化。本章只是给出了涉及用这些方法可能的应用的一些简单知识。然而，通过使用你在这里学到的术语，你应该能够学到区分每天都在发展的许多进步之间的细微差别。

既然我们已经花了一些时间学习了从简单到复杂的许多不同类型的预测模型，那么在下一章中，我们将开始思考用于其他类型的学习任务的方法。这些无监督的学习技术将会带来数据内部的引人入胜的模式。

探寻模式——基于关联规则的购物篮分析

回想上一次你冲动性的即兴购物，或许你正在超市的结账通道等待，然后，你就顺手买了一包软口香糖或者一份单独包装的块状糖；或许，在一次深夜前往便利店买尿布和婴儿食品时，你顺便买了一瓶含咖啡因的饮料或者一箱 6 瓶装的啤酒；你甚至可能在你最喜欢的书商给你推荐这本书后，你一时兴起买了这本书。总之，不管何种情况，把口香糖和糖果放置在购物通道，把啤酒存放在尿布的旁边，以及书店似乎知道哪本书会引起你的兴趣，这一切都并非巧合。

在过去的几年中，这些类型的推荐系统都是基于营销专业人士、库存管理人员或者卖家的主观经验。而近来，机器学习已经用来研究这些购买行为的模式，条形码扫描仪、计算机库存系统以及网上购物产生的丰富的事务型数据已经用于这样的数据挖掘中。

本章介绍用来判别事务型数据中商品之间关联的机器学习方法。由于其在零售店之间的广泛使用，所以这种方法通常称为**（市场）购物篮分析（market basket analysis）**。在你完成本章学习时，你将会学到：

- 使用简单的统计性能指标，发现大型数据集中有用的关联方法。
- 如何管理事务型数据处理的特性。
- 利用关联规则对真实世界的数据进行（市场）购物篮分析所需要的完整步骤。

要了解机器学习是如何找到令人感兴趣的方案（模式）的，请继续阅读。从本章开始，你将有基础尝试自己的（市场）购物篮分析。

8.1 理解关联规则

（市场）购物篮分析的结果是一组指定商品之间关系模式的**关联规则（association rule）**。

一个典型的规则可以表述为如下的形式：

{ 花生酱, 果冻 } \rightarrow { 面包 }

这个关联规则用通俗易懂的语言来表达就是：如果购买了花生酱 (peanut butter) 和果冻 (jelly)，那么也很有可能会购买面包 (bread)。换句话说，即“花生酱和果冻意味着面包”。大括号内的一件商品或者多件商品的组合表示它们构成一个集合，或者更确切地说，就是出现在具有某种规律性的数据中的项集 (itemset)。关联规则是根据项集的子集研究得到的，例如，上述规则就是由集合 { 花生酱, 果冻, 面包 } 确定的。

基于大数据和数据库科学背景下的研究，关联规则与前面章节中所呈现的分类算法和数值预测算法不同，它不能用来进行预测，但可以用于无监督的知识发现。即便如此，你将发现关联规则学习与第 5 章中的分类规则学习有密切的联系并且有许多共同的特征。

因为关联规则学习是无监督的，所以不需要训练算法，也不需要提前标记数据。基于数据集，就可以简单地运行程序，希望探寻到令人感兴趣的关联。当然，不利的一面就是除了用从定性角度来衡量它们的可用性外（通常是某种形式的目测法来对它们进行评估），没有一个简单的方法来客观地衡量一个规则学习算法的性能。

虽然关联规则最常用于（市场）购物篮分析，但是它们对于发现许多不同类型数据的模式是有帮助的。其他的潜在应用包括：

- 在癌症数据分析中，搜寻 DNA 和蛋白质序列的有趣且频繁出现的模式。
- 查找发生在与信用卡欺诈或者保险应用相结合的购物或者医疗津贴的模式。
- 确认导致客户放弃他们的移动电话服务或者升级他们的有线电视服务套餐的行为组合。

关联规则用来搜索大量变量之间的有趣联系。人类具有这种相当直观的洞察力，但往往需要专家水平的知识或者大量的经验来以便可以在几分钟甚至几秒内实现规则学习算法。此外，有些数据过于庞大而复杂，人类要找到它们之间的联系就如同大海捞针。

用于关联规则学习的 Apriori 算法

由于事务型数据的复杂性，使得关联规则挖掘在很大程度上对于计算机和人类同样是一项具有挑战性的任务。事务型数据集通常是非常庞大的，无论是在交易的数量方面，还是在被监测的特征（即商品）数量方面。增加难度的是潜在的项集数量随着特征的数量呈指数增长，给定 k 个项，可以出现在集合中，也可以不出现在集合中，大约有 2^k 个可能的项集必须用于规则的搜索。某零售商仅销售 100 种不同的商品，大约就会有 $2^{100} = 1e+30$ 个项集需要学习算法来进行评估——一个看似不可能完成的任务。

一个灵敏的规则学习算法利用了这样一个事实：在现实中，许多潜在的商品组合极少，如果有，就在实践中发现，而不是一个接一个地评估这些项集中的每个元素。例如，尽管一个商店同时销售汽车产品和妇女的化妆品，但是集合 { 机油, 口红 } 很有可能是极其罕见的。通过忽略这些罕见（并且因此可能不太重要）的组合，就可以限制规则的搜索范围，从

而更容易管理项集的规模。

为了减少需要搜索的项集数，确定启发式算法的大部分工作已经完成。或许用规则高效地搜索大型数据库的最广泛使用的方法就称为 **Apriori** 算法，该算法由 R. Agrawal 和 R. Srikant 在 1994 年提出，并自此成为与关联规则学习有关的代名词。该名称源自这样一个事实，即该算法利用了关于频繁项集性质的一个简单的先验信念。

在进入该算法之前，值得注意的是，该算法与其他的学习算法一样，也有自身的优点和缺点。其中的一些优缺点列举如下：

优点	缺点
<ul style="list-style-type: none"> • 非常适合处理极其大量的事务型数据 • 规则中的结果很容易理解 • 对于“数据挖掘”和发现数据库中意外的知识很有用 	<ul style="list-style-type: none"> • 对于小的数据集不是很有帮助 • 需要努力地将对数据的洞察和常识区分开 • 容易从随机模式得出虚假的结论

正如前面所提到的，**Apriori** 算法采用一个简单的先验信念作为准则（指引）来减少关联规则的搜索空间：一个频繁项集的所有子集必须也是频繁的。此启发式称为 **Apriori 性质**（**Apriori property**）。通过这种敏锐的观察，能够显著地限制搜索规则的次数。例如，如果集合 { 机油, 口红 } 是频繁的，当且仅当 { 机油 } 和 { 口红 } 同时频繁地发生。因此，如果机油或者口红中只要有一个是非频繁的，那么任意一个含有这两项的集合都可以从搜索中排除。



有关 **Apriori** 算法的详细信息，请参考：Fast algorithms for mining association rule, in Proceedings of the 20th International Conference on Very Large Databases, pp. 487-499, by R. Agrawal, and R. Srikant, (1994)。

为了了解这一原理在更真实的环境中是如何应用的，我们考虑一个简单的交易数据库。下表给出了在一个虚构的医院礼品店已完成的 5 项交易：

交易号	购买的商品
1	{ 鲜花, 慰问卡, 苏打水 }
2	{ 毛绒玩具熊, 鲜花, 气球, 单独包装的块状糖 }
3	{ 慰问卡, 单独包装的块状糖, 鲜花 }
4	{ 毛绒玩具熊, 气球, 苏打水 }
5	{ 鲜花, 慰问卡, 苏打水 }

通过查看购物集合，可以推断有两种典型的购买模式。探望生病的朋友或者家人的人，往往会买一张慰问卡和气球，而探望刚生孩子的母亲的人会买毛绒玩具熊和气球。这样的模式是值得注意的，因为它们的频繁出现足以引起我们的兴趣。我们简单地运用一点儿逻辑和与该主题有关的经验就可以解释这个规则。

Apriori 算法以类似的方式应用一个项集的“趣味性”统计方法来找出更大的交易数据库中的关联规则。在下面的几节中，我们将会发现 **Apriori** 如何计算这些令人感兴趣的方法，以及它们如何结合 **Apriori** 性质来减少规则学习的次数。

1. 度量规则兴趣度——支持度和置信度

关联规则是否是令人感兴趣的，取决于两个统计量：支持度和置信度。通过为这些度量的每一个量提供最小的阈值并应用 Apriori 原则，很容易大幅度地限制报告的规则数，甚至可能会达到只有明显的规则或者常理规则。为此，仔细理解被排除在这些准则之外的规则类型是很重要的。

一个项集或者规则度量法的支持度 (support) 是指其在数据中出现的频率。例如，如前所述，项集 { 慰问卡 } { 鲜花 } 在医院礼品店数据中的支持度为 $3/5=0.6$ 。类似地，{ 慰问卡 } \rightarrow { 鲜花 } 的支持度也是 0.6。任何项集的支持度都可以计算，甚至是一个单元素的项集。例如，因为单独包装的块状糖在购物中出现的频率为 40%，所以，{ 单独包装的块状糖 } 的支持度为 $2/5=0.4$ 。定义项集 X 的支持度的函数可定义为：

$$\text{support}(X) = \frac{\text{count}(X)}{N}$$

其中， N 表示数据库中的交易次数， $\text{count}(X)$ 表示项集 X 出现在交易中的次数。

规则的置信度 (confidence) 是指该规则的预测能力或者准确度的度量，它定义为同时包含项集 X 和项集 Y 的支持度与只包含项集 X 的支持度的商：

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)}$$

从本质上讲，置信度表示交易中项或者项集 X 的出现导致项或者项集 Y 出现的比例。请记住， X 导致 Y 的置信度与 Y 导致 X 的置信度是不一样的。例如，{ 鲜花 } \rightarrow { 慰问卡 } 的置信度为 $0.6/0.8=0.75$ ，而相比之下，{ 慰问卡 } \rightarrow { 鲜花 } 的置信度为 $0.6/0.6=1.0$ 。这意味着涉及鲜花的一次购买中同时伴随着慰问卡购买的结果是 75%，而慰问卡的一次购买中同时购买鲜花的相关度为 100%。这条信息对于该礼品店的经营或许会相当有用。



你可能已经注意到支持度、置信度与第 4 章介绍的贝叶斯概率规则的相似之处。事实上， $\text{support}(A, B)$ 与 $P(A \cap B)$ 是一样的， $\text{confidence}(A \rightarrow B)$ 与 $P(B | A)$ 是一样的，只是上下文不同而已。

像 { 慰问卡 } \rightarrow { 鲜花 } 这样的规则称为强规则 (strong rule)，因为它们同时具有高支持度和置信度。发现更多强规则的一种方法就是检查礼品店中的每一个可能的商品组合，测量支持度和置信度，并只报告那些满足某种兴趣水平的规则。然而，正如前面指出的，这种策略除了最小的数据集以外，一般是不可行的。

在下一节中，你将看到 Apriori 算法如何使用基于 Apriori 原则的最低水平的支持度和置信度，通过减少规则的数量来迅速找到强规则，以达到一个更便于管理的水平。

2. 用 Apriori 原则建立规则

回想一下，Apriori 原则指的是一个频繁项集的所有子集也必须是频繁的。换句话说，如

果 $\{A, B\}$ 是频繁的，那么 $\{A\}$ 和 $\{B\}$ 都必须是频繁的。还记得，根据定义，支持度表示一个项集出现在数据中的频率。因此，如果知道 $\{A\}$ 不满足所期望的支持度阈值，那么就没有理由考虑 $\{A, B\}$ 或者任何包含 $\{A\}$ 的项集，这些项集绝不可能是频繁的。

Apriori 算法利用这个逻辑在实际评估它们之前排除潜在的关联规则，创建规则的实际过程分为两个阶段：

- 识别所有满足最小支持度阈值的项集。
- 根据满足最小置信度阈值的这些项集来创建规则。

第一阶段发生于多次迭代中，每次连续的迭代都需要评估存储一组越来越大项集的支持度。例如，迭代 1 需要评估一组 1 项的项集（1 项集），迭代 2 评估 2 项集，以此类推。每个迭代 i 的结果是一组所有满足最小支持度阈值的 i 项集。

由迭代 i 得到的所有项集结合在一起以便生成候选项集用于在迭代 $i + 1$ 中进行评估。但是 Apriori 原则甚至可以在下一轮开始之前消除其中的一些项集。如果在迭代 1 中， $\{A\}$ 、 $\{B\}$ 和 $\{C\}$ 都是频繁的，而 $\{D\}$ 不是频繁的，那么在迭代 2 中将只考虑 $\{A, B\}$ 、 $\{A, C\}$ 和 $\{B, C\}$ 。因此，该算法仅需要评估 3 个项集，而如果包含 D 的项集没有事先（a priori）消除掉，那么就需要评估 6 个项集。

保持这个想法，假设在迭代 2 过程中发现 $\{A, B\}$ 和 $\{B, C\}$ 是频繁的，而 $\{A, C\}$ 不是频繁的，尽管迭代 3 通常会从评估 $\{A, B, C\}$ 的支持度开始，但是这一步根本没有发生的必要。为什么可以不发生呢？因为子集 $\{A, C\}$ 不是频繁的，所以 Apriori 原则指出 $\{A, B, C\}$ 绝不可能是频繁的。因此，在迭代 3 中就没有生成新的项集，算法将停止。

此时，Apriori 算法的第二阶段将会开始。给定一组频繁项集，根据所有可能的子集产生关联规则。例如， $\{A, B\}$ 将产生候选规则 $\{A\} \rightarrow \{B\}$ 和 $\{B\} \rightarrow \{A\}$ 。这些规则将根据最小置信度阈值评估，任何不满足所期望的置信度的规则将被消除。

8.2 例子——用关联规则确定经常一起购买的食物杂货

正如本章引言中所指出的，（市场）购物篮分析用于许多实体商店和在线零售商的后台推荐系统。关联规则学习表明经常一起购买的商品组合在一个集合中。所学到的知识或许可能会为一家杂货连锁店优化库存、宣传促销活动或者整理店内的实际布局提供新的洞察力。例如，如果购物者经常在购买咖啡或者橙汁时，顺便购买一份早餐糕点，那么为了增加利润，商店就很有可能将糕点重新放置到离咖啡与果汁更近的地方。

在本书中，我们将根据一家杂货店的事务型数据进行一次（市场）购物篮分析。然而，该技术可以运用于许多不同类型的问题，从电影推荐，到约会地点，到寻找药物之间相互作用的危险。这样做，我们将会看到 Apriori 算法如何能够有效地评估潜在的大量关联规则。

8.2.1 第 1 步——收集数据

我们的（市场）购物篮分析将利用来自一个现实世界中的超市经营一个月的购物数据。

该数据包含了 9835 次交易，大约每天 327 次交易（在 12 小时的工作日内，大约每小时交易 30 次），这表明该零售商不是特别大，也不是特别小。



这里所用的数据改编自 Apriori R 添加包中 Groceries 数据集。有关数据集更多的信息，请参阅：Implications of probabilistic data modeling for mining association rules, in *Studies in Classification, Data Analysis, and Knowledge Organization: from Data and Information Analysis to Knowledge Engineering*, pp. 598–605, by M. Hahsler, K. Hornik, and T. Reutterer, (2006)。

在一个具有代表性的超市中，有大量不同的商品。可能有 5 种品牌的牛奶，一打不同类型的衣物洗涤剂和 3 种品牌的咖啡。鉴于零售商的大小适中，假定他们不是非常关注寻找只应用于特定品牌的牛奶或者洗涤剂的规则。考虑到这一点，所有的品牌名称就可以从购买数据中去除。这将食品杂货的数量减少到更易于管理的 169 个类型，采用大类，比如鸡肉、冷冻食品、人造黄油和汽水。



如果你希望找出特别具体的关联规则（如顾客更喜欢带有他们花生酱的葡萄果冻还是草莓果冻，你需要大量的事务型数据。大规模的连锁零售商使用数以百万计的交易数据库，以便发现特定品牌、颜色或者风格商品之间的关联。

对于哪种类型的商品有可能一起购买，你有一些猜测吗？葡萄酒和奶酪是一种常见的搭配吗？面包和黄油？茶和蜂蜜？让我们来深入挖掘数据，看看是否可以证实我们的猜测。

8.2.2 第 2 步——探索和准备数据

事务型数据与我们之前已经使用过的数据集不一样，它的存储形式稍微有点儿不同。我们之前的大部分分析所采用的数据都是矩阵形式，其中，行表示实例（案例），列表示特征。给定矩阵格式的结构后，所有的实例（案例）都要求具有完全相同的特征集。

相比较而言，事务型数据的形式更自由。与往常一样，数据中的每一行指定一个单一的实例（案例）——在本例中，为一次交易。然而，每条记录包括用逗号隔开的任意数量的产品清单，从一到许多，而不是一组特征。从本质上讲，就是实例（案例）之间的特征可能是不同的。



为了继续这样的分析，你需要从 Packt 出版社的网站下载 groceries.csv 文件，并将该文件保存到你的 R 的工作目录中。

原始的 groceries.csv 数据的前 5 行如下所示：

```
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
whole milk
pip fruit,yogurt,cream cheese,meat spreads
other vegetables,whole milk,condensed milk,long life bakery product
```

这些行表示 5 次独立的超市交易。第一次交易包括 4 种商品：citrus fruit（柑橘类水果）、semi-finished bread（半成品面包）、margarine（人造黄油）和 ready soups（即食汤），作为对比，第三次交易只包括 1 种商品：whole milk（全脂牛奶）。

假设我们尝试使用函数 `read.csv()` 加载数据，与之前分析所做的一样。R 将顺畅地读入一个矩阵形式的数据，如下表所示：

	V1	V2	V3	V4
1	citrus fruit	semi-finished bread	margarine	ready soups
2	tropical fruit	yogurt	coffee	
3	whole milk			
4	pip fruit	yogurt	cream cheest	meat spreads
5	other vegetables	whole milk	condensed milk	long life bakery product

你将注意到 R 创建了 4 个变量来存储事务型数据中的项：V1、V2、V3 和 V4。虽然 R 做到这一点很好，但是如果我们使用这样的数据，之后我们将会遇到问题。首先，R 之所以选择创建 4 个变量，是因为第一行恰好有 4 个逗号分隔开的值。但是我们知道杂货的购买可以包含 4 种以上的商品，而这些交易将不幸地被分解到矩阵的多个行中。我们可以尝试通过将具有商品数量最多的交易放到文件的顶部来解决这个问题，但是这忽略了另一个更棘手的问题。

该问题是由于通过这种方式构造数据产生的，因为 R 已经构建了一组特征，这些特征不仅记录交易中的商品，还记录这些商品出现的顺序。如果我们设想我们的学习算法是为了试图找到 V1、V2、V3 和 V4 之间的关系，那么出现在 V1 中的 whole milk（全脂牛奶）与出现在 V2 中的 whole milk（全脂牛奶）可能有不同的处理。相反，我们需要一个数据集，该数据集不会将一次交易作为一组用具体商品来填充（或者不填充）的位置，而是作为一个要么包含要么不包含每种特定商品的市场购物篮。

1. 数据准备——为交易数据创建一个稀疏矩阵

解决此问题采用了一个称为**稀疏矩阵**（sparse matrix）的数据结构（你可能还记得在第 4 章中，我们使用了一个稀疏矩阵来处理文本数据）。类似于前面的数据集，稀疏矩阵的每一行表示一次交易，用列（即特征）表示可能出现在消费者购物篮中的每一件商品。因为在我们的杂货店数据中有 169 类不同的商品，所以我们的稀疏矩阵将包含 169 列。

为什么不像我们在大多数分析中所做的那样，将其存储为一个数据框呢？其原因就是，一旦增加额外的交易和商品，传统的数据结构很快就会变得过大而导致内存不足。即使这里使用的事务型数据集相对较小，但是矩阵包含了将近 170 万个单元（元素），其中大部分单元（元素）为零（因此命名为“稀疏”矩阵）。因为存储的所有这些零值没有益处，所以稀疏矩阵实际上在内存中没有存储完整的矩阵，只是存储了由一个商品所占用的单元（元素），这使得该结构的内存效率比一个大小相当的矩阵或者数据框的内存效率更高。

为了根据事务型数据创建稀疏矩阵的数据结构，可以使用由关联规则（arules）添加包提供的函数。使用命令 `install.packages("arules")` 与命令 `library(arules)`

安装和加载添加包。



关于 `arules` 添加包更多的信息，请参考：`arules -- A computational environment for mining association rules and frequent item sets`, Journal of Statistical Software Vol. 14 by M. Hahsler, B. Gruen, and K. Hornik, (2005)。

我们将采用类似于 `read.csv()` 的 `read.transactions()` 函数，该函数可以产生一个适用于事务型数据的稀疏矩阵，而 `read.csv()` 函数则不能。参数 `sep=","` 指定输入到文件中的项之间用逗号隔开。为了将 `groceries.csv` 数据读入到一个名为 `groceries` 的稀疏矩阵中，请输入：

```
> groceries <- read.transactions("groceries.csv", sep = ",")
```

如果想查看我们刚刚创建的 `groceries` 数据集的一些基本信息，可对该对象使用 `summary()` 函数：

```
> summary(groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146
```

输出信息中的第一块（如上所示）提供了一个我们创建的稀疏矩阵的概要。`9835rows` 指的是商店的交易，`169columns` 指的是可能出现在消费者购物篮中的 169 类不同商品的每一类的特征。如果在相对应的交易中，该商品被购买了，则矩阵中的该单元格为 1，否则为 0。

密度（density） 值 0.026 091 46（2.6%）指的是非零矩阵单元的比例。因为矩阵中有 $9835 \times 169 = 1\,662\,115$ 个位置，所以可以计算在商店经营的 30 天内，共有 $1\,662\,115 \times 0.026\,091\,46 = 43\,367$ 件商品被购买（假设没有重复的商品被购买）。进一步，我们可以确定平均交易包含了 $43\,367 / 9835 = 4.409$ 种不同的杂货商品（当然，如果我们多往下看一看输出信息，我们将会看到，该结果已经为我们计算好了）。

`summary()` 输出的下一块（如下所示）列出了事务型数据中最常购买的商品。因为 $2513 / 9835 = 0.2555$ ，所以我们可以确定 `whole milk`（全脂牛奶）有 25.6% 的概率出现在交易中，位于清单上的其他常见商品有 `other vegetables`（其他蔬菜）、`rolls/buns`（面包 / 馒头）、`soda`（汽水）和 `yogurt`（酸奶）。

```
most frequent items:
      whole milk other vegetables      rolls/buns
      2513          1903          1809
      soda          yogurt          (Other)
      1715          1372          34055
```

最后，它为我们呈现了一组关于交易规模的统计。总共有 2159 次交易只包含一件单一的商品，而有一次交易包含了 32 类商品，第一四分位数和中位数的购买规模分别为 2 类商品和 3 类商品，这意味着 25% 的交易包含了两件或者更少的商品，大约一半的交易中的商品数在 3 类左右，均值 4.409 与我们手动计算的值是一致的。

```

element (itemset/transaction) length distribution:
sizes
  1    2    3    4    5    6    7    8    9   10   11   12
2159 1643 1299 1005  855  645  545  438  350  246  182  117
 13   14   15   16   17   18   19   20   21   22   23   24
 78   77   55   46   29   14   14    9   11    4    6    1
 26   27   28   29   32
  1    1    1    3    1

      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   2.000   3.000  4.409   6.000  32.000

```

`arules()` 添加包包含了一些用于检查交易数据的有用功能。使用 `inspect()` 函数与向量运算的组合，可以查看稀疏矩阵的内容。前 5 项交易如下所示：

```

> inspect(groceries[1:5])
items
1 {citrus fruit,
   margarine,
   ready soups,
   semi-finished bread}
2 {coffee,
   tropical fruit,
   yogurt}
3 {whole milk}
4 {cream cheese,
   meat spreads,
   pip fruit,
   yogurt}
5 {condensed milk,
   long life bakery product,
   other vegetables,
   whole milk}

```

这些交易符合我们所查看的原始 CSV 文件。若要研究一件特定的商品（即一列数据），可以使用 `row`、`column` 矩阵概念。同时与 `itemFrequency()` 函数一起使用，我们可以看到包含该商品的交易比例。例如，这使得我们可以看到 `groceries` 数据中前 3 件商品的支持度：

```

> itemFrequency(groceries[, 1:3])
abrasive cleaner artif. sweetener  baby cosmetics
0.0035587189      0.0032536858      0.0006100661

```

注意，稀疏矩阵中商品所在的列是按字母表的顺序排序的。`abrasive cleaner`（擦洗剂）和 `artif. sweetener`（人造甜味剂）大约以 0.3% 的比例出现在交易中，而 `baby cosmetics`（婴儿用品）大约以 0.06% 的比例出现在交易中。

2. 可视化商品的支持度——商品的频率图

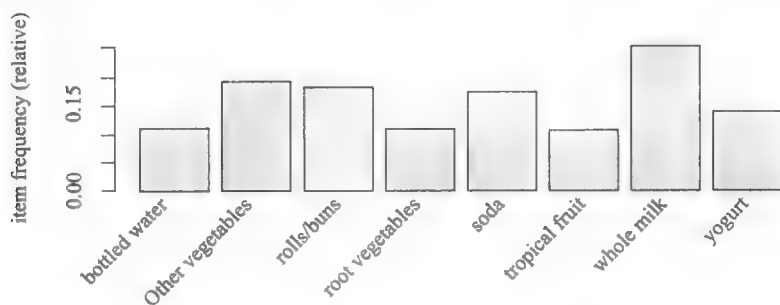
为了直观地呈现这些统计数据，可使用 `itemFrequencyPlot()` 函数，这可以让你

生成一个用于描绘所包含的特定商品的交易比例的柱状图。因为事务型数据包含了非常多的项，所以你将时常需要限制出现在图中的项，以便产生一幅清晰的图。

如果你希望获得那些出现在最小交易比例中的商品，那么可以在 `itemFrequencyPlot()` 函数中运用 `support` 参数：

```
> itemFrequencyPlot(groceries, support = 0.1)
```

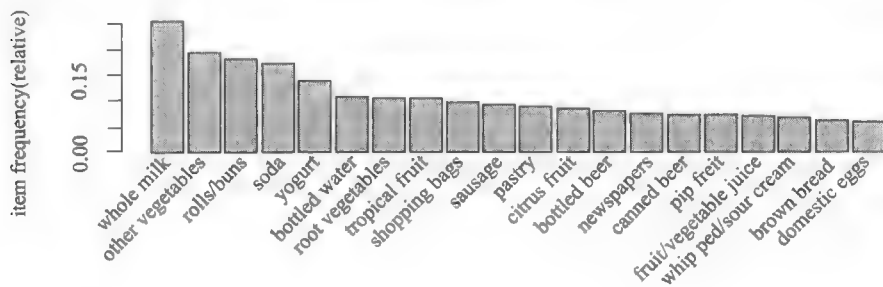
如下图所示，这生成了一个直方图，显示了 `groceries` 数据中支持度至少为 10% 的 8 类商品。



如果你更愿意限制图中商品的具体数量，那么可以在 `itemFrequencyPlot()` 函数中使用 `topN` 参数：

```
> itemFrequencyPlot(groceries, topN = 20)
```

然后直方图根据支持度降序排列，下图显示了 `groceries` 数据中的前 20 类商品：

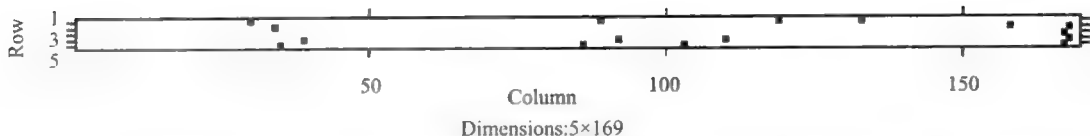


3. 可视化交易数据——绘制稀疏矩阵

除了可视化商品，也可以可视化整个稀疏矩阵。要做到这一点，需要使用 `image()` 函数。前 5 次交易的稀疏矩阵如下：

```
> image(groceries[1:5])
```

生成的图描绘了一个 5 行 169 列的矩阵，表示我们要求的 5 次交易和 169 类可能的商品。矩阵中填充有黑色的单元表示在此次交易（行）中，该商品（列）被购买了。



尽管这个图形很小，阅读起来会稍微有些困难，但是你可以看到第一次、第四次和第五次交易各包含了 4 类商品，因为它们所在的行有 4 个单元被填充了。你也可以看到在第三行、第五行、第二行和第四行有一类共同的商品（在图的右侧）。

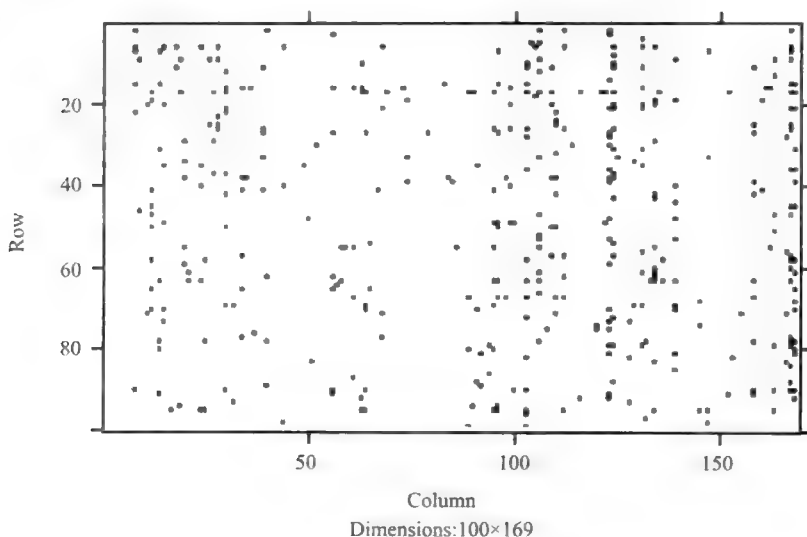
这种可视化是用于探索数据的一种很有用的工具。一方面，它可能有助于识别潜在的数据问题——列从上往下一直被填充可能表明这些商品在每一次交易都被购买了——但如果一个零售商名称或者它们的标识号意外地包含在一个数据集中，或许就会产生一个问题。

另一方面，图中的模式可能有助于揭示交易或者商品的有趣部分，特别是当数据以有趣的方式排序后。例如，如果交易按日期排序，那么黑色圆点的图案可能会揭示人们购买商品的数量或者类型的季节性影响。或许在圣诞节（Christmas）或者光明节（Hanukkah）前后，玩具的购买更常见；或许在万圣节（Halloween）前后，糖果变得更受欢迎。如果商品也被分类，那么这种类型的可视化可能会特别有效。然而，在大多数情况下，图形看上去都是相当随机的，就像电视荧屏上的静电一样。

请记住这种可视化对于超大型的交易数据集是没有用的，因为单元会太小而无法辨别。不过，通过将其与 `sample()` 函数结合，你可以看到稀疏矩阵中一组随机抽样的交易。下面看起来像是随机选择的 100 次交易：

```
> image(sample(groceries, 100))
```

这样就产生了一个 100 行 169 列的矩阵图，如下图所示。



少数列的黑点看起来是相当稠密的，表明该商店里有一些非常受欢迎的商品，但总体来说，点的分布还是相当随机的。鉴于没有其他的说明，那么就让我们继续分析吧。

8.2.3 第3步——基于数据训练模型

随着数据准备工作的完成，现在我们可以致力于寻找购物车中商品之间的关联。我们将使用在探索和准备 groceries 数据中一直使用的 arules 添加包中的 Apriori 算法实现。如果你还没有安装和加载这个添加包，那么你需要先安装和加载该添加包。下面显示了用 `apriori()` 函数来创建规则集的语法。

关联规则语法
应用 arules 添加包中的函数 <code>apriori()</code>
<p>找出关联规则：</p> <pre>myrules <- apriori(data = mydata, parameter = list(support = 0.1, confidence = 0.8, minlen = 1))</pre> <ul style="list-style-type: none"> • <code>data</code>: 含有交易数据的稀疏矩阵 • <code>support</code>: 给出要求的最低规则支持度 • <code>confidence</code>: 给出要求的最低规则置信度 • <code>minlen</code>: 给出要求的规则最低项数 <p>该函数将返回一个满足最低准则要求的规则对象。</p> <p>检验关联规则：</p> <pre>inspect(myrules)</pre> <ul style="list-style-type: none"> • <code>myrules</code> 是由函数 <code>apriori()</code> 函数给出的一组关联规则 <p>它将输出关联规则到屏幕。可以对 <code>myrules</code> 应用向量运算来选择查看一个或者多个特定的规则。</p> <p>例子：</p> <pre>concrete_model <- neuralnet(strength ~ cement + slag + ash, data = concrete) model_results <- compute(concrete_model, concrete_data) strength_predictions <- model_results\$net.result</pre>

虽然运行 `apriori()` 函数很简单，但是当找到支持度和置信度参数来产生合理数量的关联规则时，有时候可能需要进行大量的试验与误差评估。如果你将这些参数设置过高，那么你可能会发现没有规则或者规则过于普通而不是非常有用。另一方面，阈值太低可能会导致规则的数量庞大，或者更糟糕的是，该算法可能需要很长的运行时间或者在学习阶段耗尽内存。

既然这样，如果我们试图使用默认的设置: `support = 0.1` 和 `confidence = 0.8`，我们最终将不能得到任何规则：

```
> apriori(groceries)
set of 0 rules
```

显然，我们需要放宽一点搜索范围。



如果你仔细想想，这样的结果应该就不会那么令人惊奇。采用值为 0.1 的默认支持度，就意味着为了产生一个规则，一种商品必须至少出现在 $0.1 \times 9835 = 983.5$ 次交易中。而在我们的数据中，只有 8 类商品出现得比较频繁，因此我们没有发现任何规则也不足为奇。

解决支持度设定问题的一种方法是在考虑一个有趣的模式之前，事先想好需要的最小交易数量。例如，你可以认为如果一种商品一天被购买了两次（大约 60 种商品），那么这或许值得考虑看看。据此，可以计算所需要的支持度，至少可以发现仅与你事先想好的那么多次交易相一致的规则。因为 $60/9835 = 0.006$ ，所以我们将尝试首先设定支持度为 0.006。

设定最小置信度涉及一个巧妙的平衡。一方面，如果置信度太低，就可能会被大量不可靠的规则淹没（比如数 10 种规则表明与电池一起经常被购买的商品），那么我们如何知道广告预算目标呢？另一方面，如果将置信度设置得太高，那么我们将会被显而易见或者不可避免的规则所限制（比如，烟雾探测器总是与电池一起组合购买的事实）。在这种情况下，因为两类商品几乎总是一起购买，所以将烟雾探测器移到离电池更近的地方难以再产生额外的收入。



合适的最小置信度水平的选取，绝大部分取决于你的分析目标。如果你以保守值开始时，假如你没有发现具有可行性的规则，那么你总是可以降低要求以拓宽规则的搜索范围。

我们将从置信度阈值 0.25 开始，这意味着为了将规则包含在结果中，此时规则的正确率至少为 25%。这将消除最不可靠的规则，同时为我们有针对性的营销修正行为（问题）提供了一些空间。

现在，我们准备生成一些规则。除了最小支持度和置信度外，设定 `minlen = 2` 有助于消除包含少于两类商品的规则。这可以防止仅仅是由于某商品被频繁购买而创建的无趣规则（例如，`{ } => whole milk`）。此规则满足最小支持度和置信度，因为 whole milk（全脂牛奶）的购买超过 25% 的交易，但它不是一个非常可行的规则。

使用 Apriori 算法寻找一组关联规则的完整命令如下所示：

```
> groceryrules <- apriori(groceries, parameter = list(support =
  0.006, confidence = 0.25, minlen = 2))
```

该命令用 `rules` 对象保存规则，我们可以通过输入它的名称来查看：

```
> groceryrules
set of 463 rules
```

`groceryrules` 对象包含了一组 463 个关联规则。为了确定它们是否是有用的，我们必须深入挖掘。

8.2.4 第 4 步——评估模型的性能

为了获取高级关联规则的概览，可以使用如下所示的 `summary()`。规则的长度分布

(length distribution) 告诉我们按商品的每一类计数的规则有多少。在我们的规则集中, 有 150 个规则只包含 2 类商品, 而有 297 个规则包含 3 类商品, 有 16 个规则包含 4 类商品, 而与此分布相关的统计量也给出来了, 如下所示:

```
> summary(groceryrules)
set of 463 rules

rule length distribution (lhs + rhs): sizes
  2   3   4
150 297  16

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
2.000  2.000   3.000   2.711  3.000   4.000
```



如上面的输出所指出的, 规则的规模(大小)是由规则的前项(条件项或左项, lhs)与规则的后项(结果项或右项, rhs)相加得到的。这意味着, 规则 {bread} => {butter} 为 2 项, 规则 {peanut butter, jelly} => {bread} 为 3 项。

接下来, 我们看一看用于度量规则质量的主要统计量: 支持度 (support)、置信度 (confidence) 和提升度 (lift)。因为我们使用支持度和置信度作为规则的选择标准, 所以这两个统计量应该不会令人感到很惊讶。然而, 如果大多数或者所有的规则都非常接近最小阈值(这里的案例不是这种情况), 那么我们可能会感到震惊。

```
summary of quality measures:

support      confidence      lift
Min.   :0.006101  Min.   :0.2500  Min.   :0.9932
1st Qu.:0.007117  1st Qu.:0.2971  1st Qu.:1.6229
Median :0.008744  Median :0.3554  Median :1.9332
Mean   :0.011539  Mean   :0.3786  Mean   :2.0351
3rd Qu.:0.012303  3rd Qu.:0.4495  3rd Qu.:2.3565
Max.   :0.074835  Max.   :0.6600  Max.   :3.9565
```

第 3 列提升度 (lift) 是新引入的一个度量标准。假设你知道另一类商品已经被购买, 提升度就是用来度量一类商品相对于它的一般购买率, 此时被购买的可能性有多大。它的定义如下面的公式所示:

$$\text{lift}(X \rightarrow Y) = \frac{\text{confidence}(X \rightarrow Y)}{\text{support}(X)}$$



与置信度的商品购买顺序问题不同, $\text{lift}(X \rightarrow Y)$ 与 $\text{lift}(Y \rightarrow X)$ 是相同的。

例如, 假设在一家杂货店, 大多数人都购买 milk (牛奶) 和 bread (面包)。只通过偶然的机, 我们期望找到同时购买 milk 和 bread 的许多交易。然而, 如果 $\text{lift}(\text{milk} \rightarrow \text{bread})$ 大于 1, 那么这就意味着偶然发现这两类商品在一起比发现只有一类商品更常见。因此, 一

个大的提升度值是一个重要的指标，它表明一个规则是很重要的，并反映了商品之间的真实联系。

在 `summary()` 输出的最后部分，得到的挖掘信息告诉我们如何选择规则。这里，我们看到包含 9835 次交易的 `groceries` 数据，用来构建最小支持度为 0.06 与最小置信度为 0.25 的规则：

```
mining info:
      data ntransactions support confidence
groceries      9835      0.006      0.25
```

我们可以使用 `inspect()` 函数看一看具体的规则。例如，对象 `groceryrules` 中的前 3 个规则如下所示：

```
> inspect(groceryrules[1:3])

  lhs      rhs      support confidence  lift
1 {potted plants} => {whole milk}      0.006914082  0.4000000 1.565460
2 {pasta}      => {whole milk}      0.006100661  0.4054054 1.586614
3 {herbs}      => {root vegetables} 0.007015760  0.4312500 3.956477
```

由 `lhs` 和 `rhs` 表示的列指的是规则的前项（条件项或者左项）（Left-Hand Side, LHS）和后项（结果项或者右项）（Right-Hand Side, RHS）。LHS 表示为了触发规则需要满足的条件，而 RHS 表示满足条件后的预期结果。

第一条规则可以用通俗易懂的语言来表示“如果一个顾客购买了 `potted plants`（盆栽植物），那么他还会购买 `whole milk`（全脂牛奶）。”其支持度（`support`）大约为 0.007，置信度（`confidence`）为 0.400，我们可以确定该规则涵盖了大约 0.7% 的交易，而且涉及 `potted plants`（盆栽植物）购买的正确率为 40%。提升度（`lift`）值告诉我们假定一个顾客购买了 `potted plants`（盆栽植物），他相对于一般顾客会购买 `whole milk`（全脂牛奶）的可能性有多大。因为我们知道大约有 25.6% 的顾客购买了 `whole milk`（全脂牛奶）（支持度），而购买 `potted plants`（盆栽植物）的顾客有 40% 购买了 `whole milk`（全脂牛奶）（置信度），所以我们可以计算提升度为 $0.40/0.256=1.56$ ，这与显示的结果是一致的（注意，标有 `support` 的列表示规则的支持度，而不是 `lhs` 或者 `rhs` 的支持度）。

尽管规则 `{potted plants} => {whole milk}` 的置信度和提升度都很高，但是该规则看起来像是一个非常有用的规则吗？或许不是——为什么会有人在购买 `potted plants`（盆栽植物）时，更可能购买牛奶呢？似乎没有一个合乎逻辑的理由。然而，我们的数据表明并非如此，我们怎样才能使得这一事实是有意义的呢？

一种常见的做法就是获取学习关联规则的结果，并把它们分成 3 类：

- 可行的规则。
- 平凡的规则。
- 令人费解的规则。

显然，（市场）购物篮分析的目标就是要找到**可行的**（actionable）规则或者能够提供明确的有益启示的规则。有些规则是明确的，有些规则是有用的，而找到包含这两个因素的规则

是不太常见的。

平凡的 (Trivial) 规则包括那些过于明显以至于不值一提的规则——它们是很明确的，但不是很有用。假设你是一个被支付了大笔资金为交叉推广（促销）的商品确定新机遇的营销顾问，如果你报告的结论为： $\{\text{diapers}\} \Rightarrow \{\text{formula}\}$ （{ 尿布 } \Rightarrow { 配方奶 }），那么你很有可能不会再被邀请回来做另外一份咨询工作。



平凡的规则也可以伪装成更有趣的结果。例如，假设你发现了某一特定品牌的儿童谷物（麦片）和某一特定 DVD 影片之间的关联。但是如果该影片的男主角位于谷物（麦片）盒的前面，那么该发现并不能算是令人感兴趣的。

如果商品之间的规则过于不明确以至于搞清楚如何使用这些信息采取行动需要更多的研究，那么这样的规则就是**令人费解的 (inexplicable)**。这样的规则可能仅仅只是数据中的一种随机模式，例如，规则 $\{\text{pickles}\} \Rightarrow \{\text{chocolate ice cream}\}$ （{ 泡菜 } \Rightarrow { 巧克力冰淇淋 }），可能只是由于一个单一的顾客，其怀孕的妻子定期渴望奇怪组合的食物。

最好的规则是隐藏的宝石——那些未被发现的模式中的深刻见解，一旦被发现，似乎就很明显。只要有足够的时间，一个人就可以评估 463 个规则中的每一个规则以发现宝石。然而，对于一个规则是可行的、平凡的，还是令人费解的，我们（进行（市场）购物篮分析的人员）可能不是最佳的裁判。在下一节中，我们将采用方法对所学的规则进行排序和分配以提高我们工作的效用，从而使得最有趣的结果浮现出来。

8.2.5 第 5 步——提高模型的性能

主题专家 (subject matter experts) 可能能够很快找出有用的规则，但要求他们评估数百条甚至数千条规则将会浪费他们很多时间。因此，能够根据不同的标准对规则进行排序，并将它们从 R 中提取出来，形成可以与营销团队共享，而且可以进行深入探讨的形式是很有用的。这样，我们就可以通过使结果更加可行来提高规则的性能。

1. 对关联规则集合排序

根据（市场）购物篮分析的目标，最有用的规则或许是那些具有最高支持度、置信度和提升度的规则。arules 添加包包含一个 `sort()` 函数，可以用来对规则列表重新排序，而那些具有最高或者最低质量度量值的规则将会排在第一位。

要重新排列 `groceryrules`，我们可以应用 `sort()`，同时指定参数 `by` 为 "support"、"confidence" 或者 "lift"。通过将排序与向量运算相结合，可以获得指定数目的有趣规则，例如，使用下面的命令，根据提升度统计量，可以研究最好的 5 个规则：

```
> inspect(sort(groceryrules, by = "lift")[1:5])
```

这看起来将像下面的屏幕截图：

	lhs	rhs	support	confidence	lift
1	{herbs}	=> {root vegetables}	0.007015760	0.4312500	3.956477
2	{berries}	=> {whipped/sour cream}	0.009049314	0.2721713	3.796886
3	{other vegetables, tropical fruit, whole milk}	=> {root vegetables}	0.007015760	0.4107143	3.768074
4	{beef, other vegetables}	=> {root vegetables}	0.007930859	0.4020619	3.688692
5	{other vegetables, tropical fruit}	=> {pip fruit}	0.009456024	0.2634561	3.482649

这些规则似乎比我们之前看到的规则更令人感兴趣。第一条规则，提升度为 3.956 477，这意味着购买 herbs（药草）的顾客比一般顾客有将近 4 倍的可能性购买 root vegetables（根菜类蔬菜）——或许是用于某种类型的汤吗？第二条规则同样令人感兴趣。相对于其他的购物车，whipped cream（鲜奶油）有超过 3 倍的可能性在具有 berries（浆果）的购物车中被发现，这或许表明是一种甜点搭配吗？



在默认情况下，排序的顺序是递减的，这意味着最大值排在第一位，为了反转这一排序，可添加另一个参数 `decreasing = FALSE`。

2. 提取关联规则的子集

假设给定上述规则，营销团队对于有可能创建一个广告来促销正处于旺季的 berries（浆果）感到非常激动。然而，在落实广告活动之前，他们要求你调查 berries（浆果）是否经常与其他商品一起购买。要回答这个问题，我们需要找到以某种形式包含 berries（浆果）的所有规则。

`subset()` 函数提供了一种用来寻找交易、商品（项）或者规则子集的方法。要用该函数在所有规则中寻找那些包含 berries 的所有规则，通过使用下面的命令，将把（满足条件的）规则存储在一个名为 `berryrules` 的新对象中：

```
> berryrules <- subset(groceryrules, items %in% "berries")
```

当我们已经处理完较大的集合时，我们可以查看这些满足条件的规则：

```
> inspect(berryrules)
```

结果为如下所示的规则集：

	lhs	rhs	support	confidence	lift
1	{berries}	=> {whipped/sour cream}	0.009049314	0.2721713	3.796886
2	{berries}	=> {yogurt}	0.010574479	0.3180428	2.279848
3	{berries}	=> {other vegetables}	0.010269446	0.3088685	1.596280
4	{berries}	=> {whole milk}	0.011794611	0.3547401	1.388328

涉及 berries（浆果）的规则有 4 个，其中有 2 个似乎很令人感兴趣，足以称为可行的规则。除了 whipped cream（鲜奶油）外，berries（浆果）也经常与 yogurt（酸奶）一起购买——一种可以作为很好的早餐或者午餐以及甜点的搭配。

`subset()` 函数是非常强大的，选择子集的标准可以用几个关键词和运算符来定义：

□ 前面解释过的关键词 `items`，与出现在规则任何位置的项相匹配。为了将子集限制

到匹配只发生在左侧或者右侧的位置上，可使用 `lhs` 和 `rhs` 代替。

- ❑ 运算符 `%in%` 意味着至少有一项在定义的列表中可以找到。如果想得到要么与 `berries` (浆果) 相匹配，要么与 `yogurt` (酸奶) 相匹配的规则，那么可以输入 `items %in% c("berries", "yogurt")`。
- ❑ 用于部分匹配 (`%pin%`) 和完全匹配 (`%ain%`) 的额外的运算符是可用的。部分匹配允许使用一次搜索 (`items %pin% "fruit"`)，就可以找到既包含 `citrus fruit` (柑橘类水果) 又包含 `tropical fruit` (热带水果) 的规则。完全匹配需要所有列出的项都存在，例如，`items %ain% c("berries", "yogurt")` 只能找到既包含 `berries` (浆果) 又包含 `yogurt` (酸奶) 的规则。
- ❑ 子集同样可以用支持度、置信度和提升度来加以限制，例如，`confidence > 0.50` 将限制你只能找到置信度大于 50% 的规则。
- ❑ 匹配准则可以与 R 中标准的逻辑运算符 (比如，与 (&)、或 (|) 和非 (!)) 相结合。使用这些选项，可以限制你想要选择的规则为特定的规则或者是一般的规则。

3. 将关联规则保存到文件或者数据框中

若要分享 (市场) 购物篮分析的结果，你可以使用 `write()` 函数将规则保存到 CSV 文件中。这将产生一个可以在大多数电子表格程序 (包括 Microsoft Excel) 中使用的 CSV 文件：

```
> write(groceryrules, file = "groceryrules.csv",
       sep = ",", quote = TRUE, row.names = FALSE)
```

有时候将其转换成 R 中的数据框也是很方便的。使用 `as()` 函数，可以很容易实现，如下所示：

```
> groceryrules_df <- as(groceryrules, "data.frame")
```

这样就创建了一个数据框，其中规则是因子 (factor) 格式，支持度 (support)、置信度 (confidence) 和提升度 (lift) 为数值向量：

```
> str(groceryrules_df)
'data.frame':   463 obs. of 4 variables:
 $ rules      : Factor w/ 463 levels "{baking powder} => {other
vegetables}",...: 340 302 207 206 208 341 402 21 139 140 ...
 $ support    : num  0.00691 0.0061 0.00702 0.00773 0.00773 ...
 $ confidence : num  0.4 0.405 0.431 0.475 0.475 ...
 $ lift       : num  1.57 1.59 3.96 2.45 1.86 ...
```

如果你想对规则进行进一步的处理或者你需要将它们导入另一个数据库，你可能会选择这么做。

8.3 总结

关联规则是解决大数据问题的一种方法。作为一种无监督的学习算法，它们能够从没有

任何关于模式的先验知识的大型数据库中提取知识。美中不足的是，将大量的信息缩减成更小、更易于管理的结果集需要一些努力。而本章研究的 Apriori 算法可以通过设置兴趣度的最小阈值和只呈现满足这些准则的关联来解决这个问题。

在我们对一个中等规模超市的一个月的交易价值进行（市场）购物篮分析时，我们使用了 Apriori 算法。即使在这样一个小案例中，我们还是发现了大量的关联。在这些关联中，我们注意到有些模式对于未来的营销活动可能很有用。这里使用的方法同样适用于更大的零售商，它们的数据库规模可能是本例中数据库规模的好几倍。

在下一章中，我们将研究另外一种无监督的学习算法，该算法与关联规则一样，目的也是探寻数据中的模式，但其与探寻关联规则的内部特征模式不同，下一章中的方法更加关注找到案例之间的联系。



寻找数据的分组——k 均值聚类

你曾经有过花时间观看大量人群的行为吗？作为社会学家，这是我最喜欢的消遣方式之一。我会选择一个人来人往的地段（比如，咖啡厅、图书馆或者自助餐厅）来观察大量人群的有趣的行为模式，目的是为了寻找人们相互之间以及与他们周围环境相联系的一般规律的细节。

你进行这样的观察研究越多，你可能看到反复出现的个体也会越多。或许根据一套刚熨过的西服和一个公文包，就可以确定这种类型的人是典型的商务白领高管。一个 20 岁出头，穿着紧身牛仔裤和法兰绒衬衫，戴着太阳镜的小伙可能属于赶时髦的一类人；而一个让孩子从小型货车上下车的女人可能会被贴上“足球妈妈”（soccer mom）的标签。

当然，将这些类型的老套观念应用于个人是很危险的——没有哪两个人是完全一样的。然而，在用于聚类时，这些标签可能反映了属于同一组内的个体之间的一些潜在的相似模式。

本章介绍了用于处理聚类任务的机器学习方法，其中包括找到数据的自然分组。正如你将看到的，聚类的过程与刚刚描述的观察研究的过程是非常相似的。在本章中，你将学习：

- 不同于我们先前研究的分类任务的聚类任务方法以及聚类如何定义分组。
 - k 均值的基本方法，它是一种经典的并且容易理解的聚类算法。
 - 如何将聚类应用到现实世界的市场细分任务，例如青少年社交媒体用户的市场细分。
- 在采取行动之前，让我们从深入了解聚类到底能够带来什么开始。

9.1 理解聚类

聚类是一种无监督的机器学习任务，它可以自动将数据划分成类（cluster），或者具有类似项的分组。因此，聚类分组不需要提前被告知所划分的组应该是什么样的。因为我们甚至可能都不知道我们在寻找什么，所以聚类是用于知识发现而不是预测。它提供了一种从数据

内部发现自然分组的深刻洞察。

如果没有预先了解一个组是由什么构成的，那么一台计算机如何可能知道到哪里一组结束了，而另一组开始了呢？答案很简单。聚类的原则是一个组内的记录，彼此必须非常相似，而与该组之外的记录截然不同。正如你稍后将看到的，遍及所有的应用，相似的定义都可能会不同，但是基本的思想总是相同的：对数据分组时，总是使得相关的元素放在一起。

然后，所得到的结果就可以用于行动。例如，你可能会发现如下的一些应用会采用聚类方法：

- 用客户细分为具有相同的人口地理特征或者相同购买模式的组别，从而应用于有针对性的营销活动，或者根据组别来对购买行为进行详细分析。
- 通过识别使用模式不同于已知组别来检测异常行为，比如侦测未经授权入侵计算机网络的行为。
- 将大量具有相似值的特征划分成较小的具有同质类特征的组中，从而简化特大的数据集。

总之，聚类是非常有用的，无论何时，各种数据集都可以用较小的多个数据组来表示。而且，通过聚类生成的有意义的和可行的数据内部结构降低了数据的复杂性，并且提供了关于关系模式的深刻见解。

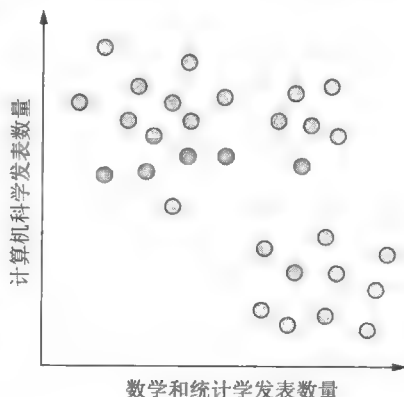
9.1.1 聚类——一种机器学习任务

聚类与我们目前为止已经研究过的分类、数值预测和模式检测任务稍有不同。在我们研究过的这些案例中，每一个案例的结果都是一个特征与结果或者特征与其他特征相关的模型。这些模型可以用来识别数据内部的模式。相比之下，聚类创建了新数据。给无标签案例一个类标签，并完全根据数据的内部关系进行推断。由于这个原因，你有时将会看到聚类称为**无监督分类** (unsupervised classification)，因为从某种意义上讲，这就是对无标签案例进行分类。

美中不足的是，从无监督分类器获得的分类标签没有内在的意义。聚类将告诉你样本中的哪些类别是密切相关的（例如，它可能会返回类 *A*、*B* 和 *C*），但应用一个可行的且有意义的类标签是由你来决定的。为了了解这个问题如何影响聚类任务，让我们来考虑一个虚假的例子。

假设你正在组织一个以数据科学为主题的会议。为了便于形成专业网络及协作，你打算根据他们的研究专长：计算机与数据库科学、数学与统计学和机器学习 3 个研究专业分组，安排大家到不同的组就坐。不幸的是，在你发送了会议邀请函之后，你意识到你忘记在邀请函中包含一份调查以询问与会者更愿意就座于哪个学科组中。

然而，灵光一闪，你意识到或许可以通过研究他的论文发表历史来推断每位学者的研究专业。为此，你开始收集数据，关于每一位与会者在计算机科学相关杂志上发表的论文数量以及在数学或者统计相关杂志上发表的论文数量。使用所收集的几位学者的数据，你创建了一个散点图：

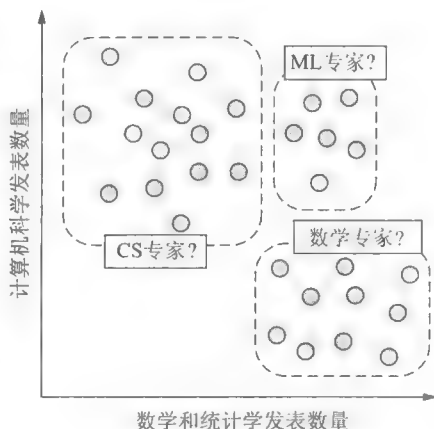


正如预期的那样，这里似乎有一种模式。我们可以猜测左上角可能是由计算机科学家构成的一个类，因为左上角表示发表过许多计算机科学论文而少量关于数学论文的学者。按照这一逻辑，右下角可能是由数学家构成的类。类似地，右上角可能是机器学习专家，因为右上角表示那些既有数学经验又有计算机科学经验的学者。

与主观地定义分组的边界相比，使用机器学习方法来客观地定义分组更好。在上图中给定平行于坐标轴的分割线，这个问题看起来就像是第5章中描述的决策树应用。这将为我们提供一个清晰的规则，如“如果某位学者只发表过少量的数学论文，那么他是计算机科学家”。遗憾的是，这个计划有一个问题，因为我们不知道每个点的真实分类值，所以我们无法采用有监督的学习算法。

我们形成的分组只是视觉上的，我们简单地根据靠近的数据分组来确定了类。尽管是看似明显的分组，但是如果如果没有亲自询问每一位学者关于他的学术专长，我们还是没有办法知道他们是否真正地属于同一类。我们应用的标签要求我们对落入组内的学者的研究类型进行定性假定判断。因此，你可以用右图中所示的不确定的术语来假定类的标签。

聚类算法的过程与我们通过视觉查看散点图所做的有些类似。通过使用一个度量样本相关程度有多紧密的度量，就可以将它们分配到同质的组中。在下一节中，我们将开始研究聚类算法是如何实现的。



这个例子突出强调了聚类的一个有趣应用。如果你一开始从无标签的数据入手，那么你就可以使用聚类来创建分类标签。然后，你可以应用一个有监督学习算法（比如，决策树）来寻找这些类中最重要的预测指标，这称为半监督学习 (semi-supervised learning)。

9.1.2 k 均值聚类算法

k 均值算法可能是最常用的聚类方法，该算法已经被研究了几十年，是许多更加复杂聚类技术的基础。如果你理解了该算法使用的简单原则，你将会拥有理解当今使用的几乎所有聚类算法的知识。许多这样的方法都列在下面的网站（有关聚类的 CRAN 任务视图）中：

<http://cran.r-project.org/web/views/Cluster.html>



随着时间的推移，k 均值算法也在发生演变，因此关于该算法，有许多不同的实现。一种流行的方法的介绍可参见：A k-means clustering algorithm in *Applied Statistics*, Vol. 28, pp. 100-108, by Hartigan, J.A. and Wong, M.A. (1979).

尽管自 k 均值算法提出以来，聚类方法已经有了改进，但这并不意味着 k 均值算法已经

过时。事实上，该方法现在可能比以前更受欢迎。下表列出了一些为什么 k 均值算法仍然广泛使用的原因。


优点	缺点
<ul style="list-style-type: none">● 使用简单的原则来确定可以用非统计术语解释的类● 它具有高度的灵活性，并且通过简单的调整，它就可以进行修正以克服它几乎所有的缺点● 将数据划分成有用的类，它是相当有效的且运行良好	<ul style="list-style-type: none">● 相对于更多近期的聚类算法，它不够复杂● 因为它使用了一个随机的元素，所以它不能保证找到最佳的类● 需要一个合理的猜测：数据有多少个自然类

如果 k 均值这个名称对你来说听起来有点儿熟悉，你可能会想起第 3 章中介绍的 k 近邻算法。你很快就会看到， k 均值与 k 近邻有很多共同点，不仅仅是字母 k 。

k 均值算法涉及将 n 个案例中的每一个案例分配到 k 个类中的一个，其中 k 是一个提前定义好的数，其目标是最小化每一个类内部的差异，最大化类之间的差异。

除非 k 和 n 是极小的，否则遍及案例所有可能的组合，计算最优的聚类是不可行的。取而代之，该算法使用了一个可以找到局部最优解的启发式过程。简单来说，这意味着它从类分配的最初猜测开始，然后对分配稍加修正以查看该变化是否提升了类内部的同质性。

很快，我们会深入讨论这个过程，然而该算法本质上包括两个阶段。首先，该算法将案例分配到初始的 k 个类中。其次，根据落入当前类的案例调整类的边界来更新分配。重复更新和分配过程多次，直到做出的改变不会再提升类的优度。至此，该过程停止，聚类最终确定。

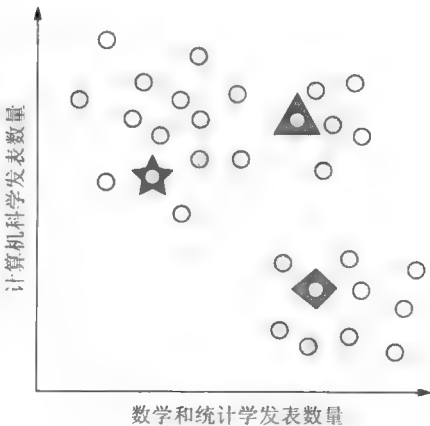


由于 k 均值的启发式性质，你可能只需要对初始条件做出轻微的改变，就能以略有不同的最终结果结束。如果结果相差很大，这可能表示存在问题。例如，该数据可能不存在自然分组或者已经选定的 k 值很差。为此，尝试不止一次的聚类分析来测试研究结果的稳健性或许是一种很好的想法。

为了了解在实际中分配和更新过程是如何运作的，让我们来重新审视用于数据科学会议的个案例数据。虽然这是一个简单的例子，但它说明 k 均值在后台是如何运作的基本原理。

1. 使用距离来分配和更新类

就像 k 近邻一样， k 均值将特征值作为一个多维特征空间中的坐标。对于数据科学会议的数据，因为只有两个特征，所以可以将特征空间表示为如先前所描述的一个 2 维散点图。 k 均值算法首先在特征空间中选择 k 个点作为类中心。这些中心是促使剩余的案例落入特征空间中的催化剂（触发因素）。通常这些点是根据从训练数据集中选择的 k 个随机案例而确定的。因为我们希望确定 3 个类，所以选中 $k=3$ 个点。这些点在下图中由星形、三角形和菱形表示。



还有一些其他的方法可以用来选择初始的类中心。一种方法是选择发生在特征空间任意地方的随机值（而不是只在数据的观测值之间进行选择）；另一种方法是完全跳过这一步，通过将每个案例随机分配到一个类中，该算法可以直接跳过这一阶段，立即进入更新阶段。这里的每一种方法都会对最终的聚类结果产生一个特定的偏差，或许你可以使用它来调整结果。

在选择了初始类中心之后，其他的案例将分配到与其最相似或者根据距离函数最相近的类中心。你会想起在我们学习 k 近邻的时候，我们研究过距离函数。通常，k 均值使用欧氏距离 (Euclidean distance)，但是有时候也使用曼哈顿距离 (Manhattan distance) 或者闵可夫斯基距离 (Minkowski distance)。回想一下，如果 n 表示特征的数量，那么案例 x 和案例 y 之间的欧氏距离公式如下为：

$$\text{dist}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

例如，如果我们将一位发表过 5 篇计算机科学论文和 1 篇数学论文的会议来宾与一位没有发表过计算机科学论文但发表过 2 篇数学论文的会议来宾进行比较，我们可以在 R 中进行的计算：

```
> sqrt((5 - 0)^2 + (1 - 2)^2)
[1] 5.09902
```

使用该距离函数，我们可以计算每一案例与每一个类中心之间的距离。然后，该案例就会被分配给离它最近的类中心。

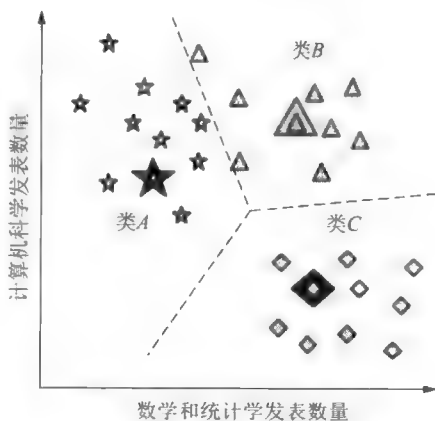


请记住，因为我们使用距离计算，所以所有的数据都必须是数值型的，而且所有的值都需要提前标准化到一个标准范围内。第 3 章介绍的方法在这里将会证明很有帮助。

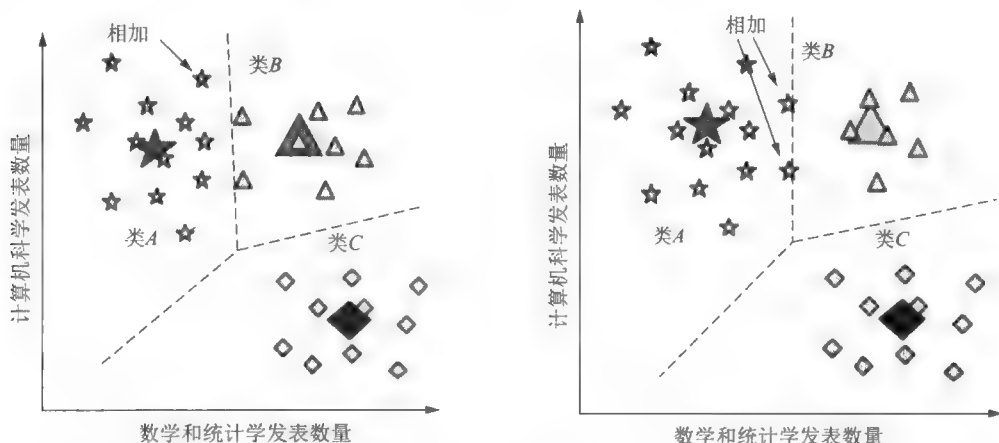
如下图所示，3 个类中心将案例划分到标有类 A、类 B 和类 C 的 3 个区域中。虚线表示由类中心创建的沃罗诺伊图 (Voronoi diagram) 的边界。沃罗诺伊图给出相对于任何其他类中心更接近于当前的类中心的区域。3 条类边界汇合的顶点是到 3 个类中心的距离最大的点。利用这些边界，我们可以很容易看到由每一个初始的 k 均值种子所确定的区域：

既然初始的分配阶段已经完成，则 k 均值算法就转到更新阶段。更新类的第一步涉及将初始的类中心转移到一个新的位置，称为质心 (centroid)，它可以通过计算分配到当前类中的各点的均值来得到。左下图说明了类中心如何变换到新的质心。

因为类的边界已经根据重新定位的中心进行了



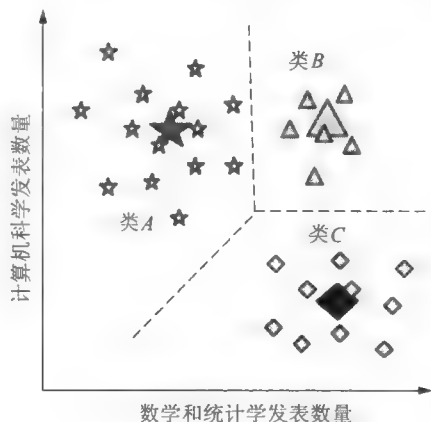
调整, 所以类 A 中可以分配到来自类 B 中的额外的案例 (如图中箭头所示)。由于这种重新分配, 所以 k 均值算法将继续进入下一个更新阶段。在对类重新计算质心之后, 得到的图如右下图所示。



在这个阶段中, 又有两个点从类 B 中重新分配到类 A 中, 因为现在这两个点相对于类 B 的质心, 更接近于类 A 的质心。这样就导致了另一个更新阶段, 如下图所示。

由于没有点在此阶段被重新分配, 所以 k 均值算法停止。到目前为止, 聚类分配最终完成。

聚类学习的最终结果可以通过以下两种方式之一来表达。一方面, 你可以简单地报告每一个案例的分类情况; 另一方面, 你也可以报告在最后一次更新之后, 聚类质心的坐标。给定两种表达方法中的任意一种, 你能够通过计算质心或者将每一个案例分配到离它最近的类中来定义类的边界。



2. 选择适当的聚类数

在 k 均值的介绍中, 我们了解到该算法对于随机选择的聚类中心很敏感。事实上, 如果在前面的例子中, 我们选择了一个关于 3 个初始点的不同组合, 我们可能会发现划分数据的类与我们之前所预期的不一样。



选择类的数目需要一种微妙的平衡, 设定非常大的 k 会提升类的同质性, 但与此同时, 会有过度拟合数据的风险。

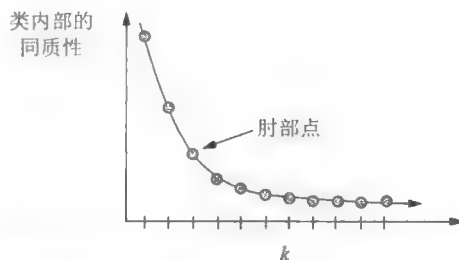
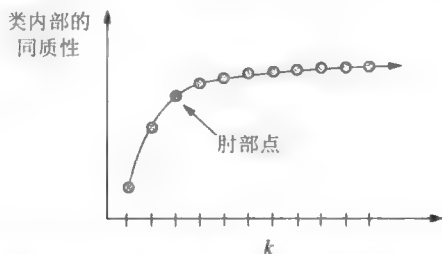
理想情况下, 你将有一些关于真实分组的先验知识 (即先验信念), 使用该信息, 你可以开始应用 k 均值算法。例如, 如果你对电影聚类, 你可能会从设置 k 等于奥斯卡奖所考虑的

体裁数开始。在我们之前所研究的数据科学会议的就坐问题中， k 可能表示被邀请的学术研究领域的数量。

有时候，聚类数是由业务需求或者分析动机所决定的。例如，在会议大厅中的表格数量决定了需要根据数据科学会议与会者清单创建多少组学者。将这种思想拓展到业务案例中，如果营销部门只有用来创建 3 个不同活动的资源，那么设定 $k=3$ 来将所有潜在的客户分配到 3 个有吸引力活动的任意一个可能就是有意义的。

如果没有任何先验知识，一个经验规则建议设置 k 等于 $\sqrt{n/2}$ ，其中， n 表示数据集的案例总数。然而，该经验规则可能会导致大型数据集中的聚类数比较庞大。幸运的是，还有其他的统计方法可以帮助找到合适的 k 均值聚类集。

称为肘部法 (elbow method) 的技术试图度量对于不同的 k 值，类内部的同质性或者异质性是如何变化的。如下图所示，随着额外的类的加入，类内部的同质性应该是期望上升的；类似地，异质性也将随着越来越多的类而持续减小。因为你可以持续地看到改进直到每个案例在一个由他自己构成的类中，所以我们的目标不是最大化同质性或者最小化异质性，而是要找到一个 k ，使得高于该值之后的收益会发生递减。这个 k 值称为肘部点 (elbow point)，因为它看起来像一个人的肘部。



有许多用来度量类内部同质性和异质性的统计量可以与肘部法一起使用。然而，在实践中，反复测试大量的 k 值并不总是可行的。部分原因是因为对大型数据集进行聚类相当费时，而对数据进行重复聚类则会更加糟糕。不管怎样，要求获得类最优解集的应用是相当罕见的。在大多数聚类应用中，基于方便选择一个 k 值就足够了，而不需要基于严格的性能要求来选择 k 值。



对于大量度量聚类性能方法的全面回顾，可查看：On clustering validation techniques, Journal of Intelligent Information Systems Vol. 17, pp. 107-145, by M. Halkidi, Y. Batistakis, and M. Vazirgiannis (2001)。

设定 k 值本身的过程有时候可能会导致有趣的见解。通过观察当 k 值变化时，类的特征是如何改变的，就可以推断出数据自然地定义边界的地方。比较紧密集聚的组变化不会太大，而具有较少同质的组将会先形成，然后随着时间的推移而解散。

一般情况下，在设定正确 k 值上不要花费太多时间可能是明智的。下一个例子将会演示如何把从好莱坞电影借鉴来的一点学科知识用来设定可行的 k 值，并且会发现有趣的类。因

为聚类是无监督的，所以聚类任务实际上是关于你是如何理解从算法的结果中所获得的见解的。

9.1.3 用k均值聚类探寻青少年市场细分

与朋友在社交网站（比如，Facebook 和 MySpace）上进行交互已经成为一种世界各地青少年的成年礼。这些青少年拥有相对大量的可支配收入，是令商家垂涎的一群人，商家希望向他们销售小吃、饮料、电子产品和卫生用品。

数百万的青少年对这些网站的浏览已经吸引了营销者的注意，这些营销者正在努力寻找以期在竞争日益激烈的市场中获得一席之地。获得这一竞争优势的一种方法就是确定有相似口味的青少年的细分，从而他们的客户可以避免将广告投给那些对正在销售的产品不感兴趣的青少年。例如，一种运动饮料对于那些对运动毫无兴趣的青少年很有可能是一份艰难的销售。

鉴于青少年**社交网络服务** (Social Networking Service, SNS) 页面的文字，我们可以确定有着相同兴趣（比如，体育、宗教或者音乐）的团体。聚类可以自动执行发现这一人群自然细分的进程。然而，这需要由我们来决定聚类所得到的类是否是令人感兴趣的，以及如何使用它们来做广告。让我们从头至尾来尝试一下这个过程。

9.1.4 第1步——收集数据

在本次分析中，我们将使用一个代表 30 000 名美国高中生的随机案例数据集，这些高中生于 2006 年就在知名的社交网络服务中保存个人资料。为了保护用户的匿名性，社交网络服务将保持匿名。然而，在收集数据时，对于美国的青少年来说社交网络服务（SNS）是一个很受欢迎的网站。因此，假设这些资料代表了 2006 年相当广泛的美国青少年案例是合理的。



该数据集是在圣母大学 (University of Notre Dame) 对于青少年的身份进行社会研究时编制的。如果你要将该数据用于研究目的，请引用这本书的章节。完整的数据集可以从 Packt 出版社的网站获得，其文件名为 `snsdata.csv`。为了以交互的方式继续分析，本章假定你已经将该文件保存在 R 工作目录中。

这些数据均匀采样于 4 个高中毕业年份（2006—2009）。在数据收集时的它们来自于高中四年级、三年级、二年级和一年级。使用自动化的网络爬虫，社交网络服务文件就会全文下载，并且每个青少年的性别 (gender)、年龄 (age) 以及社交网络服务上的交友数 (number of SNS friends) 都会被记录。

文本挖掘工具可用来将剩余的社交网络服务页面内容划分成单词。从出现在所有页面的前 500 个单词中，36 个单词被选来代表 5 大兴趣类，即课外活动 (extracurricular activity)、时尚 (fashion)、宗教 (religion)、浪漫 (romance) 和反社会行为 (antisocial behavior)。这 36 个单词包括足球 (football)、性感 (sexy)、亲吻 (kissed)、圣经 (bible)、购物 (shopping)、死亡

(death) 和药物 (drugs) 等单词。对每个人来说, 最终的数据集表示每个单词出现在个人社交网络服务文件中的次数。

9.1.5 第2步——探索 and 准备数据

我们可以使用 `read.csv()` 的默认设置将数据加载到数据框中:

```
> teens <- read.csv("snsdata.csv")
```

让我们也来快速看一看数据的细节。`str()` 输出的前几行如下所示:

```
> str(teens)
'data.frame': 30000 obs. of 40 variables:
 $ gradyear : int 2006 2006 2006 2006 2006 2006 2006 2006 ...
 $ gender : Factor w/ 2 levels "F","M": 2 1 2 1 NA 1 1 2 ...
 $ age : num 19 18.8 18.3 18.9 19 ...
 $ friends : int 7 0 69 0 10 142 72 17 52 39 ...
 $ basketball : int 0 0 0 0 0 0 0 0 0 0 ...
```

正如我们此前所期望的, 该数据包含了 30 000 名青少年, 其中 4 个变量表示个人特征, 36 个单词表示兴趣。

你注意到 `gender` 变量有什么奇怪的吗? 如果你仔细看, 你可能会注意到 NA 相对于值 1 和 2, 它代表的是完全不同的内容。R 用它来告诉我们该记录有一个缺失值 (missing value)——我们不知道这个人的性别 (gender)。直到现在, 我们还没有处理过缺失数据, 但是对于许多类型的分析来说, 它可能是一个很重要的问题。

让我们来看一看这个问题有多严重。一种方法就是使用 `table()` 命令, 如下所示:

```
> table(teens$gender)
      F      M
22054  5222
```

尽管这告诉我们存在的 F 和 M 值有多少个, 但是 `table()` 函数排除了值 NA, 而不是将其作为一个单独的值。为了包含值 NA (如果有的话), 只需要添加一个额外的参数:

```
> table(teens$gender, useNA = "ifany")
      F      M  <NA>
22054  5222  2724
```

这里, 我们看到有 2724 条记录 (9%) 缺失了性别 (gender) 数据。但有趣的是, 在社交网络服务数据中, 女性是男性的 4 倍多, 即男性并不像女性那样倾向于使用社交网络服务网站。

如果你查看了数据框 `teens` 中的其他变量, 你将会发现除了性别 (gender) 变量之外, 只有年龄 (age) 变量有缺失值。对于数值型数据, `summary()` 命令会告诉我们缺失值 NA 的数量:

```
> summary(teens$age)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   3.086  16.310  17.290  17.990  18.260  106.900    5086
```

对于年龄 (age) 变量, 共有 5086 条记录 (17%) 有缺失值。令人担心的一个事实是最小值和最大值似乎是不可信的, 一个 3 岁或者一个 106 岁的人就读于高中是不可能的。为了确保这些极端值对分析不会造成问题, 需要在继续讨论之前, 清除它们。

对于高中生, 一个合理的年龄范围应该包括那些至少 13 岁, 还没有超过 20 岁的学生, 任何落在这个范围之外的年龄值将会与缺失数据一样处理——我们不能相信提供的这些年龄。为了对年龄 (age) 变量重新编码, 可以使用 `ifelse()` 函数, 如果年龄 (age) 大于等于 13 岁且小于 20 岁, 就将 `teen$age` 值赋给 `teen$age`, 否则, 赋值为 NA:

```
> teens$age <- ifelse(teens$age >= 13 & teens$age < 20,
                      teens$age, NA)
```

通过复查 `summary()` 的输出, 现在我们可以看到年龄 (age) 的范围服从一个看上去更像一个真实高中学生的分布:

```
> summary(teens$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  13.03  16.30   17.26   17.25   18.22   20.00   5523
```

不幸的是, 现在已经导致了一个更大的缺失数据问题。在继续分析之前, 需要找到一种方法来处理这些缺失值。

1. 数据准备——缺失值的虚拟编码

一种简单的处理缺失值的方法就是排除具有缺失值的记录。然而, 如果你全面地考虑过这种做法的负面影响, 在这样做之前, 你可能需要慎重 (我这里说这种方法很简单, 但是我从来没说过这是一个好的方法)。该方法的问题在于即使缺失值不是太多, 你就非常快速地排除了大部分数据。

例如, 假设在数据中, 性别变量的值为 NA 的学生与那些缺失年龄 (age) 数据的学生是完全不同的。这将意味着通过排除那些要么缺失性别 (gender) 值, 要么缺失年龄 (age) 值的记录, 你将排除 26% 的数据, 即 9% 与 17% 的和 ($9\%+17\%=26\%$), 超过 7500 条记录, 而这是在只有两个变量有缺失数据的情况下。数据集中存在的缺失值的数量越多, 任意给定的记录被排除的可能性就越大。很快, 你将只剩下一个数据的微小子集, 或者更糟糕的是, 剩余的记录系统性的不同于总体或者不能代表总体。

对于分类数据像性别 (gender) 这样的变量, 另一种解决方法就是将缺失值作为一个单独的类别。例如, 除了局限在限制为女性 (female) 和男性 (male) 两个取值外, 我们可以增加一个额外的水平值 “unknown” (未知)。与此同时, 我们还应该利用虚拟编码, 这在第 3 章中有过深入的介绍, 就是将名义的性别 (gender) 变量转化为可以用于距离计算的数值形式的变量。

除了有一个水平值被拿出来作为参照组以外, 虚拟编码涉及为名义特征中的每个水平值单独创建一个取值为 1 或者 0 的二元虚拟变量。有一个类水平值被排除在外, 因为它

可以通过其他类的水平值来进行推断。例如，如果有人性别既不是女性 (female) 也是未知 (unknown) 的，则他们的性别一定是男性 (male)。因此，我们只需要为女性 (female) 和未知 (unknown) 的性别创建虚拟变量：

```
> teens$female <- ifelse(teens$gender == "F" &
                        !is.na(teens$gender), 1, 0)
> teens$no_gender <- ifelse(is.na(teens$gender), 1, 0)
```

第一个语句表示如果性别等于 F 且不等于 NA，则 `teens$female` 赋值为 1，否则赋值为 0。第二个语句表示 `is.na()` 函数用来检测性别 (gender) 是否等于 NA，如果 `is.na()` 返回 TRUE，则 `teens$no_gender` 赋值为 1，否则赋值为 0。为了证实我们所做的工作是正确的，让我们将构建的虚拟变量与原始的 `gender` 变量进行比较：

```
> table(teens$gender, useNA = "ifany")
  F      M  <NA>
22054  5222  2724
> table(teens$female, useNA = "ifany")
  0      1
7946 22054
> table(teens$no_gender, useNA = "ifany")
  0      1
27276  2724
```

因为在 `teens$female` 和 `teens$no_gender` 中值为 1 的数量与初始编码中 F 和 NA 值的数量是一致的，所以我们应该可以信任上述的工作。

2. 数据准备——插补缺失值

接下来，让我们来消除关于年龄 (age) 变量的 5523 个缺失值。因为年龄 (age) 是数值型的，所以给未知值创建一个额外的类别是没有意义的——那么相对于其他的年龄 (age) 值，你将 “unknown” (未知) 排在哪儿呢？取而代之，我们将使用一种称为插补法 (imputation) 的不同策略，插补法依据可能的真实值的猜测来填补缺失值。

你是否能想到一种方法，从而我们能够使用社交网络服务数据使得一个受过教育的个人可以对一个青少年的年龄做出猜测吗？如果你想到使用毕业的年份，那么你已经有了好的想法，在一个毕业生队列中，大部分学生都是在同一年内出生的。如果我们能够找出每一个队列中具有代表性的年龄，那么我们将对那年毕业的学生的年龄有一个相当合理的估计。

找到具有代表性值的一种方法就是通过计算平均值。如果我们像之前的分析中所做的那样尝试应用 `mean()` 函数，就会有一个问题：

```
> mean(teens$age)
[1] NA
```

问题在于，对包含缺失数据的向量，其均值是无法定义的。因为年龄 (age) 包含缺失值，所以 `mean(teens$age)` 返回一个缺失值。在计算均值之前，可以通过添加一个额外的参数来去除缺失值，从而修正这个问题：

```
> mean(teens$age, na.rm = TRUE)
[1] 17.25243
```

这表明在我们的数据中，学生的平均年龄大约为 17 岁。这里我们只是达到了部分目的，我们真正需要的是每一个毕业年份的年龄平均值。你也许会想去计算 4 次均值。但 R 的优点是它通常有一种更有效的方式。在这种情况下，`aggregate()` 函数是完成这项工作的工具，它可以为数据的子组计算统计量。这里，在去除 NA 值后，它计算毕业年份（`gradyear`）的不同水平值的年龄均值：

```
> aggregate(data = teens, age ~ gradyear, mean, na.rm = TRUE)
  gradyear      age
1    2006 18.65586
2    2007 17.70617
3    2008 16.76770
4    2009 15.81957
```

毕业年份每变化大约一年，平均年龄就会不同。这一点并不令人惊讶，而一个有用的发现证明了我们的数据是合理的。

`aggregate()` 的输出在一个数据框中，它是人们可以读取的，但是还需要额外的工作来把它合并到我们的原始数据中：作为一种替代方法，可以使用 `ave()` 函数，该函数返回一个具有重复的组均值的向量，使得结果在长度上等于原始向量的长度：

```
> ave_age <- ave(teens$age, teens$gradyear, FUN =
  function(x) mean(x, na.rm = TRUE))
```

为了将这些均值插补到缺失值中，需要再一次使用 `ifelse()` 函数，仅当原始的年龄（`age`）值为 NA 时，调用 `ave_age` 的值：

```
> teens$age <- ifelse(is.na(teens$age), ave_age, teens$age)
```

`summary()` 的结果表明现在已经消除了缺失值：

```
> summary(teens$age)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 13.03   16.28   17.24   17.24   18.21   20.00
```

随着用于分析的数据已经准备好，我们开始准备深入到该项目的有趣部分。让我们来看看前面的努力是否已见成效。

9.1.6 第 3 步——基于数据训练模型

为了将青少年进行市场细分，我们使用 `stats` 添加包中的一个 `k` 均值实现，该添加包应该包含在 R 的默认安装中。如果碰巧你没有这个包，你可以像安装任何其他添加包一样安装该添加包，并使用 `library(stats)` 命令来加载这个包。尽管 `k` 均值函数在不同的 R 添加包中并不缺乏，但是 `stats` 添加包中的 `kmeans()` 函数是广泛使用的，并且它提供了一个平凡的算法实现。

聚类语法

应用 stats 添加包中的函数 kmeans()

建立模型:

```
myclusters <- kmeans(mydata, k)
```

- mydata: 是包含需要聚类的实例的一个矩阵或者数据框
- k: 给定需要的类的个数

该函数返回一个含有聚类结果的聚类对象

检验聚类结果:

- myclusters\$cluster 是 kmeans() 函数所给出的类成员向量
- myclusters\$centers 是含有每个类组合和每一个特征的均值的一个矩阵
- myclusters\$size 给出每一个类中实例的个数

例子:

```
teen_clusters <- kmeans(teens, 5)
teens$cluster_id <- teen_clusters$cluster
```

kmeans() 函数需要一个只包含数值型数据的数据框和一个用来指类的期望数目的参数。如果你已经准备好这两部分,那么真正建立模型的过程是很简单的。麻烦的是选择数据和类的正确组合是一门艺术,有时候会陷入到大量的试验和错误中。

为了开始聚类分析,我们将只考虑 36 个特征,这些特征代表出现在社交网络服务青少年资料中的不同兴趣数。为了方便,让我们来创建一个只包含这些特征的数据框:

```
> interests <- teens[5:40]
```

在使用距离计算分析之前,通常采用的做法是将特征标准化或者 z-score 标准化以便使得每个特征具有相同的尺度。如果这样做,你就可以避免这样的问题,即一些特征只是因为相对于其他特征有更大的值,因而在聚类中起主导作用。

如果你回想第 3 章, z-score 标准化就是重新调整特征以使得它们的均值为 0, 标准差为 1。这种转化从某种意义上改变了特征的解释含义,但或许在这里比较有用。特别地,如果有人在他们的个人资料中总共提到 3 次足球,那么如果没有额外的信息,我们无法知道相对于他们的同龄人来说,他们是更喜欢足球,还是更不喜欢足球。另一方面,如果 z-score 标准化后的值为 3,我们就可以知道他们提到的足球次数比一般的青少年更多。

为了将 z-score 标准化应用于数据框 interests,我们可以使用带有 lapply() 的 scale() 函数,如下所示:

```
> interests_z <- as.data.frame(lapply(interests, scale))
```



因为 lapply() 返回一个矩阵,所以它必须要使用 as.data.frame() 函数巧妙地将结果转换为数据框的形式。

我们最后的决定涉及确定使用多少类来细分数据。如果我们使用太多的类,我们可能会发现它们过于具体而没什么用;相反,选择过少的类可能会导致异质分组。使用不同的 k 值进行试验,你应该会感到轻松。如果不喜欢得到的结果,你可以很轻松地尝试另一个 k 值并

重新开始。



如果你对于人口的分析很熟悉，那么选择类的数量会比较容易；或者有对于自然分组的真实数量的预感也可以为你节省一些试验和错误。

为了帮助我们预测数据中类的数量，我会遵从一部我最喜欢的电影，即 1985 年 John Hughes 的成年喜剧，*The Breakfast Club*（早餐俱乐部）。在这部电影中，将符合年龄的高中生特征确定为 5 个方面的典型类型：聪明人（Brain）、运动员（Athlete）、没有特征（basket case）、公主（Princess）和罪犯（Criminal）。考虑到这些身份在所有流行的青少年小说中比较盛行，因此对于 k 值，5 似乎是一个相当合理的起始值。

为了将 teens 划分成 5 个类，可以使用下面的命令：

```
teen_clusters <- kmeans(interests_z, 5)
```

这样就将 k 均值聚类的结果保存到了一个名为 `teen_clusters` 的对象中。

9.1.7 第 4 步——评估模型的性能

评估聚类的结果有一定的主观性。最终，该模型的成功或者失败取决于类对于他们预期目的是否有用。因为本次分析的目标是为了确定具有相似兴趣的青少年的分类以达到营销的目的，所以我们将很大程度上从定性的方面来度量我们的成功。对于其他的聚类应用，可能需要更多的定量措施来度量成功。

评估一个类是否有用的最基本方法之一就是检查落在每一组中的案例数。如果类的案例数过多或者过少，那么这些类不太可能会有用的。为了获得 `kmeans()` 聚类的大小，可以使用 `teen_clusters$size` 分量，如下所示：

```
> teen_clusters$size
[1] 3376 601 1036 3279 21708
```

这里，我们看到了我们所要求的 5 个类，最小的类包含了 601 位（2%）青少年，而最大的类包含了 21 708 位（72%）青少年。虽然大的类有点令人担忧，但是如果没有仔细检查，我们就不会知道这是不是表示存在着问题。好消息，我们并没有发现任何只包含一个人的类，尽管这在运用 k 均值聚类时偶尔会发生。



鉴于 k 均值聚类的随机性，如果你的结果与这里显示的结果不同，不必惊慌。相反，可以认为这是一个运用你的分析技能来获得唯一结果的机会。

为了更深入地了解类，可以使用 `teen_clusters$centers` 分量来查看聚类质心的坐标，前 8 个特征的坐标如下所示：

```
> teen_clusters$centers
  basketball  football  soccer  softball
1  0.02447191  0.10550409  0.04357739 -0.02411100
2 -0.09442631  0.06927662 -0.09956009 -0.04697009
```



```

3  0.37669577  0.38401287  0.14650286  0.15136541
4  1.12232737  1.03625113  0.53915320  0.87051183
5 -0.18869703 -0.19317864 -0.09245172 -0.13366478
   volleyball      swimming cheerleading   baseball
1  0.04803724  0.31298181  0.63868578 -0.03875155
2 -0.07806216  0.04578401 -0.10703701 -0.11182941
3  0.09157715  0.24413955  0.18678448  0.28545186
4  0.78664128  0.11992750  0.01325191  0.86858544
5 -0.12850235 -0.07970857 -0.10728007 -0.13570044

```

输出的行(编号1~5)指的是类,而输出中的数值表示位于该列顶部的兴趣变量的平均值。由于所有的值已经进行了z-score标准化,所以负值表示低于所有学生的总体均值,而正值表示高于所有学生的总体均值。

虽然只给出了这8项兴趣,但是我们已经可以推断出这些类的一些特征。在所有列出来的体育运动中,除了啦啦队(cheerleading)兴趣变量之外,类4在运动项目上数值显著大于均值,表明该组可能包含运动员(athletes)。类1最有可能包含提到的啦啦队(cheerleading),并且其在兴趣项足球(football)中超过了平均水平。

通过这种方式继续研究这些类,可以构建一个表来列出每组中的主要兴趣项。在下面的图表中,显示的每一类包含最有可能与其他类区分开来的特征。有趣的是,类5被区分开是由于它的特征不显著这个事实,在每一个度量的兴趣活动中,它的特征的兴趣水平都低于平均值,但从成员人数方面来说,它又是最大的单一类。一种潜在的解释就是这些用户在网站上创建个人资料(文件),但从未发布任何兴趣爱好。

类 1 (N = 3376)	类 (2N = 601)	类 (3N = 1036)	类 (4N = 3279)	类 (5N = 21 708)
游泳 (swimming)	乐队 (band)	运动 ()	篮球 (basketball)	???
啦啦队 (cheerleading)	游行 (marching)	性 (sex)	橄榄球 (football)	
聪明 (cute)	音乐 (music)	性感 (sexy)	足球 (soccer)	
性感 (sexy)	摇滚乐 (rock)	受欢迎 (hot)	垒球 (softball)	
受欢迎 (hot)		亲吻 (kissed)	排球 (volleyball)	
跳舞 (dance)		跳舞 (dance)	运动 (sports)	
打扮 (dress)		音乐 (music)	上帝 (god)	
头发 (hair)		乐队 (band)	教堂 (church)	
商场 (mall)		筛子 (die)	基督 (Jesus)	
Hollister 牌		死亡 (death)	圣经 (bible)	
Abercrombie 牌		醉酒 (drunk)		
购物 (shopping)		吸毒 (drugs)		
服装 (clothes)				
公主 (Princess)	聪明人 (Brain)	罪犯 (Criminal)	运动员 (Athlete)	没有特征 (basket case)



当共享一个细分分析的结果时,应用能够捕捉群体本质的信息标签往往是有帮助的,就像这里用到的 The Breakfast Club (早餐俱乐部) 中的分类。而添加这种标签的风险就是他们可能通过定型组内成员来掩盖组之间的细微差别。

给定上表，一个营销执行主管将对访问社交网站的 5 种类型的青少年有一个明确的描述。基于这些个人资料，执行主管可以有针对性地向一个或多个类相关的产品的企业销售广告思想。在下一节中，为了这些用途，我们将看到如何将聚类的标签应用到原始的群体。

9.1.8 第 5 步——提高模型的性能

因为聚类创造了新的信息，所以一聚类算法的性能至少在某种程度上取决于这些类本身的质量以及如何处理这些信息。在上一节中，我们已经证明了这 5 个类提供了关于青少年兴趣的有用的并且新颖的见解。从这方面而言，聚类算法似乎表现得非常好。因此，我们现在可以集中精力将这些见解转化为行动。

首先，将把这些类应用回完整的数据集。当创建 k 均值聚类时，函数存储了一个名为 `teens$cluster` 分量，它包含了对于案例中所有 30 000 名学生的类的划分配，可以使用下面的命令，将 `teens$cluster` 作为一列添加到数据框 `teens` 中：

```
> teens$cluster <- teen_clusters$cluster
```

根据这一信息，可以确定每一位用户被分配到了哪一类中。例如，下面是社交网络服务数据中前 5 个用户的个人信息：

```
> teens[1:5, c("cluster", "gender", "age", "friends")]
  cluster gender   age friends
1       5      M 18.982        7
2       1      F 18.801         0
3       5      M 18.335        69
4       5      F 18.875         0
5       3  <NA> 18.995        10
```

使用之前我们用过的 `aggregate()` 函数，我们同样可以整体上查看这些类的人口统计特征。根据聚类，类之间每一类的年龄均值变化不大，尽管我们认为不同年龄不一定在兴趣上有系统性的差异。其描述如下所示：

```
> aggregate(data = teens, age ~ cluster, mean)
  cluster      age
1       1 16.99678
2       2 17.38765
3       3 17.10022
4       4 17.09634
5       5 17.29841
```

另一方面，根据聚类，每一类的女性比例会有一些显著的差异。这是一个有趣的发现，因为我们没有使用性别 (`gender`) 数据来进行聚类，但这些类对于性别 (`gender`) 仍然具有非常强的预测能力：

```
> aggregate(data = teens, female ~ cluster, mean)
  cluster  female
1       1 0.8942536
2       2 0.7221298
3       3 0.8001931
```

```
4      4 0.7130223
5      5 0.7109821
```

回想一下，总体上大约有 74% 的社交网络服务用户为女性。类 1（所谓的公主（Princess）组）有将近 90% 的用户为女性；而类 2、类 4 和类 5 只有大约 70% 的用户为女性。

考虑到在性别预测方面的成功，你可能也会猜想这些类对于用户拥有的朋友数量的预测能力。这个假设得到了数据的支持，如下所示：

```
> aggregate(data = teens, friends ~ cluster, mean)
  cluster friends
1      1  38.74733
2      2  32.88186
3      3  30.57046
4      4  36.14029
5      5  27.85314
```

平均来说，公主（Princess）组拥有最多的朋友（38.7），接下来是运动员（Athlet）组和聪明人（Brain）组，而罪犯（Criminal）组只有 30.6，没有特征（Basket Case）组也只有 27.9。与性别（gender）一样，因为没有将朋友的数量输入聚类算法中，所以此项发现也是很引人注目的。

组内成员身份、性别和朋友的数量之间的关系表明，这些类是非常有用的预测因子。以这种方式来验证这些类的预测能力，使得将这些类在推销给营销团队时变得更加容易，并最终提高算法的性能。

9.2 总结

我们的研究结果支持了一句流行的格言：“物以类聚，人以群分”。通过使用机器学习对具有相似兴趣的青少年进行聚类，我们能够开发一门关于青少年身份的类型学以便对个人特征（比如，性别和朋友的数量）进行预测。这些相同的方法也可以应用于其他具有相似结果的背景中。

本章只是介绍了聚类的基本原理。作为一种非常成熟的机器学习方法， k 均值算法有无数种变体，还有很多其他可以给任务带来独特见解与启发的方法。基于你在这里所学的知识，你将能够理解和运用其他聚类方法去解决新的问题。

在下一章中，我们将开始研究用于度量一种学习算法有多成功的方法，这些方法适用于许多机器学习任务。尽管在算法的应用过程中，我们一直做出努力来评估学习算法的成效，但是为了获得最高级别的性能，能够用最严格的条件来定义和度量这些学习算法还是至关重要的。



模型性能的评价

很多年以前，只有富人才能负担得起受教育的费用。那时，测验和考试并不是用来评价学生的。相反，它们用来评价老师，父母需要知道他们的孩子是否学到了足够多的知识，从而证明老师对得起他们的薪水。很显然，这种情形多年以来发生了改变。现在，类似的评价方法用来区分学生水平的高低和作为进入职业生涯或者继续深造的筛选工具。

考虑到这种方式的意义，人们在开发精确的学生评价方法上投入了大量的努力。一种公平的评价方式要包含覆盖面广泛的一些主题，要能测试出真实的知识水平而不是凭运气的猜测。这个评价方式的问题还需要能使得被测试者去思考以前没有遇到过的难题。正确的回答可以表明该学生能够将知识应用到更通用的领域。

类似于编写考试问题的过程可以用来评估机器学习算法。由于不同的算法具有不同的优点和缺点，所以在评价算法对未来数据的性能时需要使用测试来展示不同算法之间的差异。

本章将提供一些用来评价机器学习算法的信息，包括：

- 为什么预测准确度不足以度量性能，以及替代的度量性能的方法。
- 用来确保性能度量方法可以合理地反映模型对于未知数据的预测能力的方法。
- 如何使用 R 将这些更有用的度量应用到前面章节中学习的预测模型中。

你会发现，与学习某个主题的最好方法是尝试将其教给其他人一样，在评价机器学习算法的过程中也能够对如何更好地使用之前学习的算法有更深刻的理解。

10.1 度量分类方法的性能

在前面的章节中，我们使用准确度来度量分类的性能。将正确预测的部分除以预测的总

数得到的数值就是准确度。该数值表示分类器正确或者错误分类的百分比。例如，假设某个分类器能够准确地预测 100 000 个新生儿中的 99 990 个携带或者不携带某种可治疗但是可能致命的先天缺陷，那么意味着准确度为 99.99%，而错误率只有 0.01%。

虽然看上去这是一个非常精确的分类器，但是明智的做法是在放心地把孩子的生命交给这个测试之前搜集一些额外的信息。如果该种先天缺陷在每 100 000 个新生儿中只有 10 例又该怎么办？如果不考虑任何情况全部预测为“没有缺陷”，该测试仍然会有 99.99% 的准确度。既然这样的话，该分类器即使对绝大多数的数据都能预测正确，但对于其最初打算识别新生儿先天缺陷的目的却不管用。



这是类不平衡问题的影响之一，这类问题指的是当数据中有很大部分记录属于同一个类别时造成的麻烦。

对分类性能最好的度量方式要看其是否能成功地实现预期目标。因此，拥有能够度量实用性而不是原始准确度的模型性能评价方法是至关重要的。基于此目的，我们将介绍各类对来源于混淆矩阵这一熟悉格式表达的预测性能的度量方式。在开始之前，我们需要考虑如何为性能评价来准备分类的结果。

10.1.1 在 R 中处理分类预测数据

有 3 种主要的数据类型可以用来评价分类器：

- ☐ 真实的分类值。
- ☐ 预测的分类值。
- ☐ 预测的估计概率。

在之前的章节里，我们使用了前两种方法。思路是维护两个数据向量：一个用来保存真实的分类值，另一个用来保存预测的分类值。两个向量需要具有相同的长度和排列顺序。预测值和真实值可以存储为不同的 R 向量或者 R 数据框中的不同列。这两种方法都可以在大部分的 R 函数中执行。

真实的分类值直接来源于测试数据中的目标变量。例如，测试数据是一个名为 `test_data` 的数据框，目标变量是名为 `outcome` 列，可以使用类似 `actual_outcome <- test_data$outcome` 的命令来创建一个真实值的向量。

预测值使用模型来得到。在大多数机器学习的 R 添加包中，可以将 `predict()` 函数作用到某个模型对象以及测试数据的数据框来得到预测值，例如 `predicted_outcome <- predict(model, test_data)`。

到目前为止，我们只使用了数据中的这两种向量来检查分类方法的预报能力。但是隐藏在背后的是另一种有用的信息。尽管分类器对每个例子做出了统一的预测，但是在有些判断上显得更有把握。例如，一个分类器可以有 99% 的把握确信包含“免费”和“铃声”等词语的短信是垃圾信息，但是只有 51% 的把握确信包含“今晚”的短信是垃圾信息。在两种情况

下，分类器都预测成垃圾信息，但是对两者做判断的确信程度差别却很大

研究内部预测概率对于评估模型的性能非常有用，它同时也是第三种形式评估数据的来源。如果两个模型在预测时发生了相同数目的错误，但是其中一个能够更准确地估计它的不确定性，那么这个模型就是更智能的模型。理想的分类器在做出正确预测时能够非常有信心，但是在面对拿不准的情况时能够非常谨慎。在信心和谨慎之间的平衡是模型评价中的一个关键部分。

不幸的是，要得到内部预测概率是一件棘手的事情，因为对不同的分类器，计算方法是不同的。一般来说，分类器的 `predict()` 函数可以指定希望的预测类型。如果要得到单一的预测类别（例如垃圾信息或者有用信息），可以设定为 "class" 类型。如果要得到预测的概率，可以设定 `prob`、`posterior`、`raw` 或者 `probability` 等类型。



基本上本书出现的所有分类器都提供了类似的概率，具体的参数使用方法包含在介绍每个模型的语法介绍中。

例如，如果要输出朴素贝叶斯分类的预测概率，可以参考第4章的描述。我们可以在预测函数中加上参数 `type = "raw"`。例如，`predicted_prob <- predict(model, test_data, type = "raw")`。

类似地，第5章介绍的 C5.0 分类器的命令是：`predicted_prob <- predict(model, test_data, type = "prob")`。

需要记住的是，大部分情况下，`predict()` 函数将返回对结果不同水平的预测概率。例如，在结果是 “yes/no” 的二值模型中，`predicted_prob` 将是一个矩阵或者数据框，如下所示：

```
> head(predicted_prob)
      no      yes
1 0.0808272 0.9191728
2 1.0000000 0.0000000
3 0.7064238 0.2935762
4 0.1962657 0.8037343
5 0.8249874 0.1750126
6 1.0000000 0.0000000
```

在构建测试数据集时要当心，从而确保对于感兴趣的类别使用了正确的概率。为了避免混淆，在二值结果的情况下，建议在以上两个输出向量中只保留一个。

为了举例描述典型的测试数据，我们将使用第4章开发的垃圾短信分类模型判断出的估计为垃圾信息的概率以及真实类别和预测类别组成的数据框进行介绍。



如果要运行此处的例子，可以到 Packt 出版社网站下载 `sms_results.csv` 数据文件，然后使用命令 `sms_results <- read.csv("sms_results.csv")` 加载该数据框。

`sms_results` 数据框非常简单。下面是运行的命令和结果，它包含了 3 个具有 1390 个元素的向量。第一个向量表示短信的真实类别（垃圾信息（`spam`）或者有用信息（`ham`）），第二个向量表示模型的预测类别，第三个变量表示该短信是垃圾信息的概率：

```
> head(sms_results)
  actual_type predict_type   prob_spam
1         ham          ham 2.560231e-07
2         ham          ham 1.309835e-04
3         ham          ham 8.089713e-05
4         ham          ham 1.396505e-04
5        spam          spam 1.000000e+00
6         ham          ham 3.504181e-03
```

注意，当预测类型为有用信息（`ham`）时，`prob_spam` 的值非常接近于 0。相反地，当预测类型为垃圾信息（`spam`）时，`prob_spam` 的值等于 1，这意味着模型有 100% 的把握确定该条短信是垃圾信息。垃圾信息的估计概率值如此极端的事实说明了这个模型对于它的判断非常有信心。但是当预测值和真实值不同时会发生什么情况呢？使用 `subset()` 函数来找到这些记录：

```
> head(subset(sms_results, actual_type != predict_type))
  actual_type predict_type   prob_spam
53        spam          ham 0.0006796225
59        spam          ham 0.1333961018
73        spam          ham 0.3582665350
76        spam          ham 0.1224625535
81        spam          ham 0.0224863219
184       spam          ham 0.0320059616
```

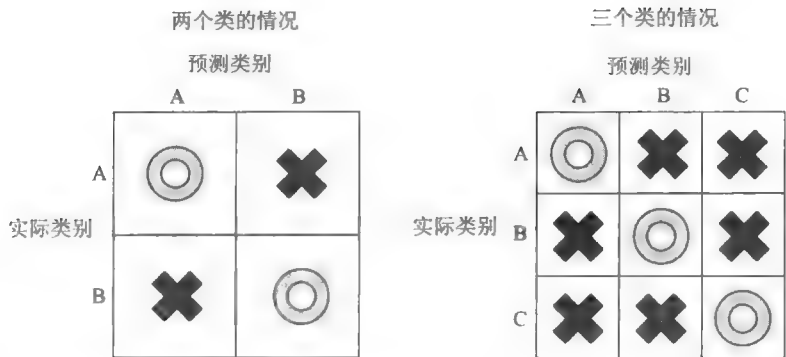
可以发现这些概率没有那么极端，尤其是第 73 行，分类器觉得它有 35% 的概率是垃圾信息，但是仍然把它归类为有用信息。

前面的 6 个例子代表了短信分类器的 6 个错误。尽管有这些错误，模型是否仍然有用？我们可以通过对测试数据应用各种误差度量的方法来回答这个问题。事实上，很多这样的度量方法都是基于我们在之前的章节里广泛使用的工具。

10.1.2 深入探讨混淆矩阵

混淆矩阵（`confusion matrix`）是一张二维表，它按照预测值是否匹配数据的真实值来对预测值进行分类。该表的第一个维度表示所有可能的预测类别，第二个维度表示真实的类别。虽然到目前为止我们只见过 2×2 的混淆矩阵，但是混淆矩阵也可以用于预测多个类别的模型。下图展示了我们熟悉的二值分类模型的混淆矩阵。对于三值分类模型，将是类似 3×3 的混淆矩阵。

当预测值和真实值相同时，就是一个正确的分类。正确的预测位于混淆矩阵的对角线上（标记为 O）。矩阵非对角线上的元素（标记为 X）表示预测值与真实值不相同的情况，它们是错误的预测。对分类模型的性能度量基于表的对角线和非对角线上预测值的个数：



最常见的模型性能度量方式主要考虑模型在所有的分类中识别出某个分类的能力。我们感兴趣的类别称为**阳性 (positive)**，其他所有类别称为**阴性 (negative)**。



对于阳性或阴性这样的术语并没有隐含任何的价值判断 (好或者坏)，同样也没有暗示出现或者不出现 (先天缺陷或者没有)。我们甚至可以把任意的类别当作阳性的结果，比如在预测类别中的晴天或雨天，狗或猫。

阳性类别的预测值和阴性类别的预测值之间的关系可以用一个 2×2 的混淆矩阵来描述。我们可以根据预测值是否落入下述 4 类中的某一个来创建这个表格矩阵：

- **真阳性 (True Positive, TP)**：正确的分类为感兴趣的类别。
- **真阴性 (True Negative, TN)**：正确的分类为不感兴趣的类别。
- **假阳性 (False Positive, FP)**：错误的分类为感兴趣的类别。
- **假阴性 (False Negative, FN)**：错误的分类为不感兴趣的类别。

在前面提到的先天缺陷的分类器中，混淆矩阵把模型预测的先天缺陷的状态是否与真实值相匹配的情况制成了表格，如下图所示。

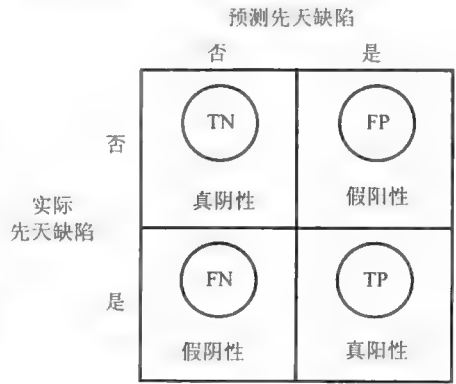
10.1.3 使用混淆矩阵度量性能

使用 2×2 的混淆矩阵，可以用公式来表示预测**准确度 (accuracy)**，有时也称为成功率)：

$$\text{准确度} = \frac{TP + TN}{TP + TN + FP + FN}$$

在这个公式中，TP、TN、FP 和 FN 指的是模型的预测值落入这些类别中的次数。因此，准确度表示真阳性和真阴性的数目除以所有预测值的个数。

错误率 (error rate)，或者说不正确分类的比例，定义如下：



$$\text{错误率} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{准确度}$$

注意，错误率可以用1减去准确度来得到。直观的理解也是有道理的，如果一个模型有95%是预测正确的，那么意味着5%是预测错误的。

有一种快捷得到混淆矩阵的方法是使用 `table()` 函数。这种方法最容易被记住，通过计算每个组合的值的数目来得到——实际上这就是我们需要的混淆矩阵。针对短信数据建立混淆矩阵的命令如下所示，表中的数目可以用来计算准确度和其他的统计量：

```
> table(sms_results$actual_type, sms_results$predict_type)
      ham spam
ham  1202   5
spam   29 154
```

如果想得到具有更详细输出的混淆矩阵，`gmodels` 添加包中的 `CrossTable()` 函数提供了高度可定制的解决方案。是否记得，第一次使用这个函数是在第2章。不过，如果没有安装该添加包，需要使用命令 `install.packages("gmodels")` 来安装它。

`CrossTable()` 的结果默认会在每一格内输出这一格的数据对表的行、列和总数的比值，同时也包含每行每列的总数。代码如下所示，其语法和 `table()` 非常类似：

```
> library(gmodels)
> CrossTable(sms_results$actual_type, sms_results$predict_type)
```

结果是包含更多详细信息的混淆矩阵：

Cell Contents			
-----N-----			
Chi-square contribution			
N / Row Total			
N / Col Total			
N / Table Total			

Total Observations in Table: 1390			
sms_results\$actual_type	sms_results\$predict_type		Row Total
	ham	spam	
ham	1202	5	1207
	16.565	128.248	
	0.996	0.004	0.868
	0.976	0.031	
	0.865	0.004	
spam	29	154	183
	109.256	845.876	
	0.158	0.842	0.132
	0.024	0.969	
	0.021	0.111	
Column Total		1231	1390
		0.886	0.114

我们在之前的几个章节中使用过 `CrossTable()` 函数，所以现在应该熟悉这里的输出

结果。如果忘记了，可以通过表格的关键词（标记为 Cell Contents）查询索引，它们对表格中每个数值的含义进行了描述。

我们可以使用该列联表（contingency table）来得到准确度和错误率。因为准确度的公式是 $(TP + TN) / (TP + TN + FP + FN)$ ，所以可以计算：

```
> (154 + 1202) / (154 + 1202 + 5 + 29)
[1] 0.9755396
```

我们同样可以根据公式 $(FP + FN) / (TP + TN + FP + FN)$ 来计算错误率：

```
> (5 + 29) / (154 + 1202 + 5 + 29)
[1] 0.02446043
```

这和 1 减去准确度的结果是相同的：

```
> 1 - 0.9755396
[1] 0.0244604
```

虽然这些计算过程看起来非常简单，但是通过练习可以更好地理解混淆矩阵中每个元素相对于其他元素的含义。在下一节中，我们将看到这些相同部分如何组合成不同的方法，提供额外的度量性能的方式。

10.1.4 准确度之外的其他性能评价指标

如果要对每一种度量性能的方法都进行全方位的描述是不现实的。数不清的度量方法被开发出来，可以用于很多学科中的特定需求，尤其是医学、信息检索、市场营销、信号检测等学科。不过，我们只关注在机器学习文献中被广泛引用的一些常见方法。

Max Kuhn 开发的分类和回归训练添加包（caret）包含了一些函数，可以用来计算很多这样的性能度量指标。该添加包提供了大量的对机器学习模型和数据准备、训练、评估以及可视化的工具。除了这里的应用以外，我们还将第 11 章对它进行更广泛的应用。在开始之前，使用 `install.packages("caret")` 命令来安装这个添加包。



如果要获取 caret 的更多信息，请参考如下文章：Building predictive models in R using the caret package, Journal of Statistical Software, Vol. 28, Iss. 5, by Max Kuhn (2008)。

caret 添加包给出了另一个创建混淆矩阵的函数。其语法和 `table()` 很相似，但是需要指定阳性的输出，具体命令如下所示。因为短信分类器的目标是检测垃圾信息，所以设置 `positive = "spam"`。

```
> library(caret)
> confusionMatrix(sms_results$predict_type, sms_results$actual_type,
positive = "spam")
```

结果如下所示：

```

Confusion Matrix and Statistics

          Reference
Prediction ham spam
ham      1202  29
spam       5  154

          Accuracy : 0.9755
          95% CI : (0.966, 0.983)
       No Information Rate : 0.8683
       P-Value [Acc > NIR] : < 2.2e-16

          Kappa : 0.8867
  Mcnemar's Test P-Value : 7.998e-05

          Sensitivity : 0.8415
          Specificity : 0.9959
         Pos Pred Value : 0.9686
         Neg Pred Value : 0.9764
          Prevalence : 0.1317
         Detection Rate : 0.1108
         Detection Prevalence : 0.1144
         'Positive' Class : spam

```

该函数的输出结果包含了混淆矩阵和其他的一些性能评价指标。让我们看一看其中比较常用的统计量。

1. Kappa 统计量

通过解释完全因为巧合而预测正确的概率，**Kappa 统计量**（在前面的输出结果中标记为 **Kappa**）对准确度进行了调整。Kappa 值的范围最大是 1，说明在预测值和真实值之间是完全一致的，这样的情况非常少见。小于 1 的值表示不完全一致。

根据模型的使用目的，对 Kappa 统计量的解释会有所不同。但是一般的解释如下所示：

- ☐ 很差的一致性：小于 0.2。
- ☐ 尚可的一致性：0.2 ~ 0.4。
- ☐ 中等的一致性：0.4 ~ 0.6。
- ☐ 不错的一致性：0.6 ~ 0.8。
- ☐ 很好的一致性：0.8 ~ 1。

但是有个问题很重要，这些类别是主观性的。如果预测某人最喜欢的冰淇淋口味，那么“不错的一致性”已经足够；如果目标是让航天飞机在月球表面安全着陆，那么“很好的一致性”也有可能还不够。



关于前面的度量方法的更多信息，请参考：The measurement of observer agreement for categorical data, Biometrics Vol. 33, pp.159-174, by J.R. Landis and G.G. Koch (1977)。

下面是计算 Kappa 统计量的公式。在这个公式中，Pr 指的是分类器和真实值之间的真实一致性 (a) 和期望一致性 (e) 的比例：

$$k = \frac{\text{Pr}(a) - \text{Pr}(e)}{1 - \text{Pr}(e)}$$



定义 Kappa 统计量有多种方法。这里描述的是最常用的方法，使用 Cohen 的 Kappa 系数，如下面论文中所述：A coefficient of agreement for nominal scales, *Education and Psychological Measurement* Vol. 20, pp. 37-46, by J. Cohen (1960)。

如果知道去哪里寻找，该比例可以非常容易地通过混淆矩阵计算得到。让我们以短信分类模型的混淆矩阵为例，使用 `CrossTable()` 函数，将之前的结果复制到这里：

sms_results\$actual_type	sms_results\$predict_type		Row Total
	ham	spam	
ham	1202	5	1207
	16.565	128.248	
	0.996	0.004	0.868
	0.976	0.031	
	0.865	0.004	
spam	29	154	183
	109.256	845.876	
	0.158	0.842	0.132
	0.024	0.969	
	0.021	0.111	
Column Total		1231	1390
		0.886	0.114

每格最底部的值表示落入这个格子的实例占有所有实例的比例。因此，要计算的一致性 $Pr(a)$ ，只需要简单地将预测值与实际 SMS 类型一致的格子的比例加起来即可。如下所示：

```
> pr_a <- 0.865 + 0.111
> pr_a
[1] 0.976
```

在这个分类器中，观测值和实际值有 97.6% 的情况是一致的，我们可以发现这个值和准确度是一样的。Kappa 统计量使用期望的一致性 $Pr(e)$ 对准确度进行调整。 $Pr(e)$ 是完全的偶然性导致的预测值和实际值相同的概率，当然需要遵循两者都是在观察到的比例下随机抽样这一假设。

为了找到这些观测比例，可以应用第 4 章中学习的概率规则。假设两个事件是独立的（意味着一个不会影响另一个），概率规则指出两者同时发生的概率等于两者各自发生概率的乘积。例如，我们知道同时选择 ham（有用信息）的概率是：

$$Pr(\text{actual_type is ham}) * Pr(\text{predicted_type is ham})$$

同时选择 spam（垃圾信息）的概率是：

$$Pr(\text{actual_type is spam}) * Pr(\text{predicted_type is spam})$$

预测值或者实际类型是 ham（有用信息）或者 spam（垃圾信息）的概率可以通过行或列的汇总得到，例如，

$$Pr(\text{actual_type is ham}) = 0.868$$

$Pr(e)$ 可以通过将“预测值与实际分类一致认为是有用信息的概率”和“预测值与实际分

类一致认为是垃圾信息的概率”相加得到。因为这两个事件是互斥事件（也就是说，它们不可能同时发生），所以我们可以简单地将两个概率相加从而得到任意一个事件发生的概率。具体的 R 代码如下所示：

```
> pr_e <- 0.868 * 0.886 + 0.132 * 0.114
> pr_e
[1] 0.784096
```

因为 $pr_e=0.784\ 096$ ，所以完全偶然的情况下观测值和实际值有 78.4% 的概率是一致的。这意味着现在我们得到了 Kappa 公式所需要的全部信息。将 pr_a 和 pr_e 放入 Kappa 的计算公式中，可以发现：

```
> k <- (pr_a - pr_e) / (1 - pr_e)
> k
[1] 0.8888395
```

Kappa 值约为 0.89，这与前面的函数 `confusionMatrix()` 的输出是一致的（细微的不同是因为舍入误差）。利用之前推荐的解释，我们可以认为该分类器在预测值和实际值中有着很好的一致性。

R 中有很多自动计算 Kappa 值的函数。可视化分类数据添加包（Visualizing Categorical Data, `vcd`）中的 `Kappa()` 函数（注意第一个字母 K 要大写）使用预测值和实际值的混淆矩阵进行计算。在使用 `install.packages("vcd")` 命令安装该添加包后，使用以下命令可以得到 Kappa 值：

```
> library(vcd)
> Kappa(table(sms_results$actual_type, sms_results$predict_type))
               value      ASE
Unweighted 0.8867172 0.01918876
Weighted   0.8867172 0.01587936
```

我们主要关注不加权的 Kappa 值，其值 0.89 符合我们的期望。



加权 Kappa 值主要用于存在不同尺度一致性的情况。例如，使用冷、温暖和热的度量方式，温暖和热之间的一致性要强于温暖和冷之间的一致性。对于二值分类的情况（比如，有用信息和垃圾信息），加权 Kappa 值和不加权 Kappa 值是一样的。

评定者间可信度（Inter-Rater Reliability, `irr`）添加包中的 `kappa2()` 函数可以直接使用数据框中的预测值向量和实际分类向量来计算 Kappa 值。通过命令 `install.packages("irr")` 安装该添加包之后，使用如下命令得到 Kappa 值：

```
> library(irr)
> kappa2(sms_results[1:2])
Cohen's Kappa for 2 Raters (Weights: unweighted)
Subjects = 1390
Raters = 2
Kappa = 0.887
```

```
z = 33.2
p-value = 0
```

这两种方法都得到了相同的 Kappa 值，所以可以任选一种觉得使用方便的方法。



注意不要使用内置的 `kappa()` 函数，它与我们之前介绍的 Kappa 统计量没有关系

2. 灵敏度与特异性

分类经常要在做决策时过于保守和过于激进之间做平衡。例如，一个邮件过滤器可以采用一种很激进的方法以错杀几乎所有正常邮件为代价确保能过滤所有的垃圾邮件。另一方面，如果要确保没有任何正常邮件被错误地过滤掉，可能要容许难以接受数目的垃圾邮件通过过滤器。这种权衡可以由灵敏度和特异性这对度量方式实现。

模型的**灵敏度**（也称为真阳性率）度量了阳性样本被正确分类的比例。因此，如以下公式所示，可以通过真阳性的数目除以数据中阳性的总数——包括正确分类的（真阳性）和错误分类的（假阴性）。

$$\text{灵敏度} = \frac{TP}{TP + FN}$$

模型的**特异性**（也称为真阴性率）度量了阴性样本被正确分类的比例。与灵敏度一样，也是通过真阴性的总数除以阴性的总数——包括真阴性和假阳性：

$$\text{特异性} = \frac{TN}{TN + FP}$$

给定短信分类器的混淆矩阵，我们可以很容易手工计算这两个指标。假设垃圾信息表示阳性类别，我们可以确认 `confusionMatrix()` 的输出中的数目是正确的。例如，灵敏度可以使用如下方式计算：

```
> sens <- 154 / (154 + 29)
> sens
[1] 0.8415301
```

类似地，计算特异性：

```
> spec <- 1202 / (1202 + 5)
> spec
[1] 0.9958575
```

`caret` 添加包提供了可以直接计算灵敏度和特异性的函数，输入预测值向量和实际分类向量即可。注意指定合适的 `positive` 和 `negative` 参数，代码如下所示：

```
> library(caret)
> sensitivity(sms_results$predict_type, sms_results$actual_type,
  positive = "spam")
```

```
[1] 0.8415301
> specificity(sms_results$predict_type, sms_results$actual_type,
             negative = "ham")
[1] 0.9958575
```

灵敏度和特异性的取值范围都是 0 ~ 1，接近 1 的值更令人满意。当然，在两者之间找到一个合适的平衡点是很重要的——这通常是由具体情况决定的。

例如，在这个例子中，灵敏度是 0.842 意味着 84% 的垃圾信息被正确分类。类似地，特异度 0.996 意味着 99.6% 的正常信息被正确分类了，或者说，0.4% 的正常信息被当作垃圾信息排除了。拒绝 0.4% 的正常信息可能让人难以接受，但这是在能有效减少垃圾信息的基础上的合理权衡。

使用灵敏度和特异性提供工具用来考虑这种权衡。典型的做法是，对模型进行调整或者使用不同的模型，直到能通过灵敏度和特异性的阈值为止。后面将要讨论的可视化方法也可以帮助理解灵敏度和特异性之间的权衡。

3. 精确度和回溯精确度

与灵敏度和特异性紧密相关的是另两个性能度量指标，同样与分类时的折中方案有关，它们是预测精确度和回溯精确度。这两个统计量最开始用于信息检索领域，目的是提供对于模型结果的有趣和有关程度的描述，或者说预测是否会因为无意义的噪声而减弱。

精确度（也称为阳性预测值）定义为真阳性在所有预测为阳性案例中的比例，换句话说，当一个模型预测阳性类别时，总是正确吗？一个精确的模型只有在类别非常像阳性时才会预测为阳性。这是非常可靠的。

我们可以考虑当模型不精确时会发生什么。经过一段时间，结果将变得不可信。在信息检索领域，这与搜索引擎类似，就好比 Google 会返回不相关的结果。最终用户将会转向其竞争对手（比如，Bing）。在垃圾信息过滤器的例子中，高预测精确度意味着模型可以很仔细地定位到垃圾信息同时会忽略有用信息。

$$\text{精确度} = \frac{TP}{TP + FP}$$

另一方面，回溯精确度是关于结果完备性的度量。在后面的公式中可以看到，它定义为真阳性与阳性总数的比例。你会发现这与灵敏度是一样的，只是解释上不一样。回溯精确度高的模型可以捕捉大量的阳性样本，这意味着其具有很宽的范围。例如，高回溯精确度的搜索引擎可能返回大量与搜索词相关的文档。类似地，如果大多数的垃圾信息被正确地识别，那么意味着垃圾短信过滤器具有较高的回溯精确度。

$$\text{回溯精确度} = \frac{TP}{TP + FN}$$

我们可以通过混淆矩阵来计算精确度和回溯精确度。同样，我们假设垃圾信息是阳性类

别，那么精确度就是：

```
> prec <- 154 / (154 + 5)
> prec
[1] 0.9685535
```

回溯精确度也可以计算得到：

```
> rec <- 154 / (154 + 29)
> rec
[1] 0.8415301
```

caret 添加包可以用来计算这两个值，只要输入预测类别和真实类别的向量。使用 `posPredValue()` 函数计算精确度：

```
> library(caret)
> posPredValue(sms_results$predict_type, sms_results$actual_type,
               positive = "spam")
[1] 0.9685535
```

使用我们之前用过的 `sensitivity()` 函数可以计算回溯精确度。

与灵敏度和特异性之间固有的权衡相似，对于大多数的真实问题，很难建立一个同时具有很高的精确度和回溯精确度的模型。实现高精确度是非常容易的，如果目标仅仅是容易摘到的果实——那些易于分类的样本。类似地，通过广泛的撒网也能很容易实现高回溯精确度，这意味着模型在预测阳性时过于激进。相比之下，同时具有高精确度和高回溯精确度非常具有挑战性。因此测试各种模型是非常重要的，这可以帮助我们找到符合当前项目需求的精确度和回溯精确度的组合。

4. F 度量

将精确度和回溯精确度合并成一个单一值的模型性能度量方式是 **F 度量**（有时也称为 F1 记分或者 F 记分）。F 度量使用调和平均值来整合精确度与回溯精确度。因为预测精确度和回溯精确度都是 0 ~ 1 之间的比例，所以使用调和平均值而不是更常用的算术平均值。以下是 F 度量的公式：

$$F \text{ 度量} = \frac{2 \times \text{精度} \times \text{回溯精确度}}{\text{回溯精确度} + \text{精度}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

使用之前计算的精确度和回溯精确度来计算 F 度量：

```
> f <- (2 * prec * rec) / (prec + rec)
> f
[1] 0.9005848
```

这与使用混淆矩阵中的计数得到数值是一样的：

```
> f2 <- (2 * 154) / (2 * 154 + 5 + 29)
> f2
[1] 0.9005848
```

因为 F 度量将模型的性能指标变成了一个单一的值，所以它提供了一种便利的方式来比较多

个模型的好坏。不过，这需要假设精确度和回溯精确度具有同样的权重，然而这个假设并不总是正确的。对精确度和回溯精确度使用不同的权重来计算 F 值是可行的。但是，在最好的情况下，选择权数是件棘手的事情，但在最坏的情况下又太过于随意。更好的实践方式是将诸如 F 度量之类的度量方式与其他更全局化考虑模型的优势和不足的方法联合起来，例如下一节介绍的方法。

10.1.5 性能权衡的可视化

可视化方法经常能帮助理解机器学习算法的性能在不同的情况下是如何不同。与成对的统计量（比如，灵敏度和特异性、精确度与回溯精确度）不同，可视化可以考察度量如何在大范围的值之间变化。它们还提供了可以在单个图形中同时比较多个分类器的方法。

ROCR 包提供了一套易于使用的函数用于可视化分类模型性能的统计量。它包含了很多函数用来计算大部分常用的性能度量指标并进行可视化。ROCR 网站（<http://rocr.bioinf.mpi-sb.mpg.de/>）列出了该添加包的所有功能，并提供了几个展现其可视化能力的例子。在继续之前，我们首先使用命令 `install.packages("ROCR")` 安装这个添加包。



关于 ROCR 开发的更详细信息，参见 *ROCR: visualizing classifier performance in R*, Bioinformatics Vol.21, pp. 3940-3941, by T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer (2005)。

使用 ROCR 创建可视化图形，需要两个数据向量。第一个必须包含预测的类别值，第二个必须包含阳性类别的估计概率。这些用来创建一个预测对象，它可以被 ROCR 的绘图函数检测到。

短信分类器的预测对象可以使用分类器的估计垃圾信息概率（`prob_spam`）和实际类别标签（`actual_type`）来生成。可以使用 `prediction()` 函数来实现：

```
> library(ROCR)
> pred <- prediction(predictions = sms_results$prob_spam,
                     labels = sms_results$actual_type)
```

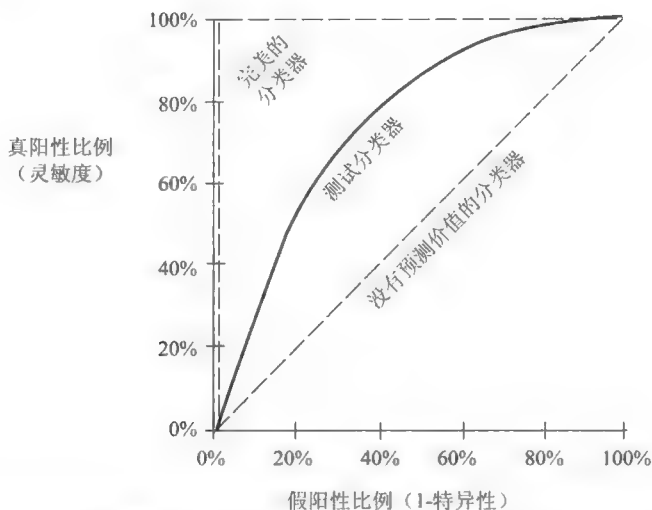
ROCR 提供了 `performance()` 函数来计算所预测对象的性能度量值，例如前面的代码示例中使用过的 `pred`。结果性能对象可以通过使用 R 中的 `plot()` 函数进行可视化。通过这 3 个函数，可以创建各种描述。

ROC 曲线

ROC（Receiver Operating Characteristic，受试者工作特征）曲线常常用来检查在找出真阳性和避免假阳性之间的权衡。通过其名称可以猜想到，ROC 曲线是第二次世界大战期间通信领域的工程师开发的。雷达和无线信号接收器需要一种方法用来区分真信号和假警报。同样的技术在今天被用来可视化机器学习模型的功效。

典型 ROC 图形的特点在下图中得到了显示。统计图形中的曲线，纵轴表示真阳性的比

例，横轴表示假阳性的比例。因为这两个值分别等于灵敏度和 1-特异性，所以该图形也称为灵敏度 / 特异性图：



ROC 曲线上的点表示不同假阳性阈值上的真阳性的比例。绘制曲线时，分类器的预测值通过模型对阳性类别的估计概率排序，最大值在最前面。从原点开始，每个预测值对应的真阳性和假阳性将导致曲线沿垂直方向（正确的预测）移动或者沿水平方向（错误的预测）移动。

为了说明这个概念，我们在上图中比较了 3 个假设的分类器。第一个是图中从左下角到右上角的直线，代表没有预测价值的分类器。这种分类器发现真阳性和假阳性的比率完全相同，这意味着该分类器无法识别两者之间的差别。这是评价其他分类器的基准线。ROC 曲线如果比较靠近这条线，则说明模型不是很有用。类似地，完美分类器拥有一条穿过了 100% 真阳性和 0% 假阳性点的曲线。它在不正确地分出任何阴性的结果之前已经正确地识别了所有的真阳性样本。大部分真实的分类器比较类似于测试分类器，它位于完美分类器和没有预测价值的（无用）分类器之间的区域

离完美分类器越接近说明能够越好地识别阳性值。可以使用 ROC 曲线下面积（Area Under the ROC, AUC）这个统计量来度量。与字面意思一样，AUC 将 ROC 图看成是 2 维正方形，然后测量 ROC 曲线下的面积。AUC 的值从 0.5（无预测值的分类器）到 1.0（完美分类器）。通常使用类似于学校字母评分的体系来解释 AUC 的得分：

- 0.9 ~ 1.0 = A（优秀）。
- 0.8 ~ 0.9 = B（良好）。
- 0.7 ~ 0.8 = C（一般）。
- 0.6 ~ 0.7 = D（很差）。
- 0.5 ~ 0.6 = F（无法区分）。

与类似的很多评分尺度一样，其水平可能在某些任务上的表现要强于其他的任务。这个分类方法比较主观。



同样值得注意的是，两个 ROC 曲线可能形状不同但是具有相同的 AUC。由于这个原因，AUC 可能具有误导性。最好的方式是在使用 AUC 的同时也对 ROC 曲线进行定性分析。

使用 ROCR 添加包绘制 ROC 曲线需要为 `pred` 预测对象构建一个性能对象，我们之前计算过它。因为 ROC 曲线是真阳性和假阳性的线图，所以可以简单地调用 `performance()` 函数来绘图，只需指定 `tpr` 和 `fpr` 度量，代码如下所示：

```
> perf <- performance(pred, measure = "tpr", x.measure = "fpr")
```

使用 `perf` 性能对象，可以使用 R 中的 `plot()` 函数绘制 ROC 曲线。如下列代码所示，很多标准参数可以用来对图形进行调节，例如 `main` (添加标题)、`col` (改变线的颜色) 和 `lwd` (调节线宽)：

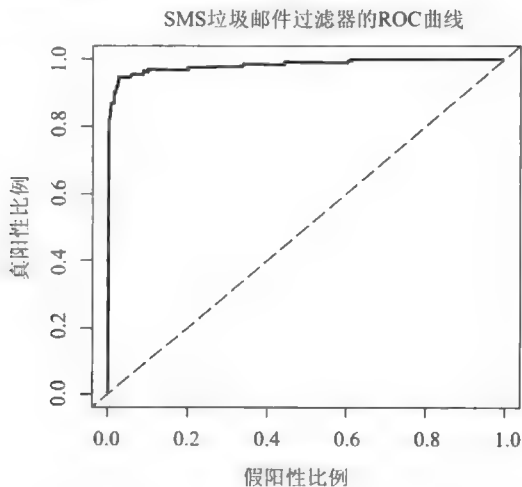
```
> plot(perf, main = "ROC curve for SMS spam filter",
      col = "blue", lwd = 3)
```

虽然代码中的 `plot()` 命令已经足够绘制 ROC 图形，但是加入一条参考线用来表示无预测值的分类器也是很有必要的。

可以使用 `abline()` 函数来绘制这条线。该函数可以通过斜率截距的方式来设定一条直线，`a` 表示截距，`b` 表示斜率。因为我们需要一条通过远点的 45 度的线，所以我们可以设置截距 `a=0`，斜率 `b=1`，如下图所示。`lwd` 参数调节线的粗细，`lty` 参数调节线型。例如，`lty=2` 表示虚线。

```
> abline(a = 0, b = 1, lwd = 2, lty = 2)
```

最终的结果是包含虚线参考线的 ROC 图：



定性地分析，我们可以看到这条 ROC 曲线占据了图形左上角的区域，这意味着与虚线代表的无用分类器相比，它更接近于完美分类器。如果要定量地确认这个事实，我们可以

使用 ROCR 添加包来计算 AUC。首先，我们需要创建另一个性能类型对象，这次我们设定 `measure = "auc"`，代码如下所示：

```
> perf.auc <- performance(pred, measure = "auc")
```

因为 `perf.auc` 是一个对象（明确地说是一个 S4 对象），所以需要使用一个特殊的符号来访问存储在其中的值。S4 对象存储信息的位置称为槽。`str()` 函数可以用来查看一个对象的所有槽：

```
> str(perf.auc)
Formal class 'performance' [package "ROCR"] with 6 slots
 ..@ x.name      : chr "None"
 ..@ y.name      : chr "Area under the ROC curve"
 ..@ alpha.name  : chr "none"
 ..@ x.values    : list()
 ..@ y.values    : List of 1
 .. ..$ : num 0.983
 ..@ alpha.values: list()
```

注意，槽的前缀是符号 @。要想访问 AUC 的值，其作为列表对象存储在 `y.values` 槽内，可以使用符号 @ 和 `unlist()` 函数，将列表简化成数值向量：

```
> unlist(perf.auc@y.values)
[1] 0.9829999
```

短信分类器的 AUC 是 0.98，非常高。但是我们如何才能知道这个模型对于其他的数据集是否也表现得足够好？为了回答这个问题，我们需要更好地理解在测试数据之外我们能够推断模型的预测性能。

10.2 评估未来的性能

有些 R 机器学习添加包在模型构建过程中显示混淆矩阵和性能度量指标。这些统计量的目的是提供对模型再带入误差的认识，虽然模型直接从数据中构建，但是当训练数据进行了错误的预测时就会产生再带入误差。该信息用于一个粗略的诊断工具，尤其是用于识别明显不好的分类器。

但是对于未来的性能，再带入误差并不是很好的标识器。例如，如果一个模型通过死记硬背的方式对每个训练的个体都进行了完美分类（再带入误差为 0），那么对于它从来没有见过的数据将无法进行预测。为此，训练数据的误差率对于模型的未来性能可能会过于乐观。

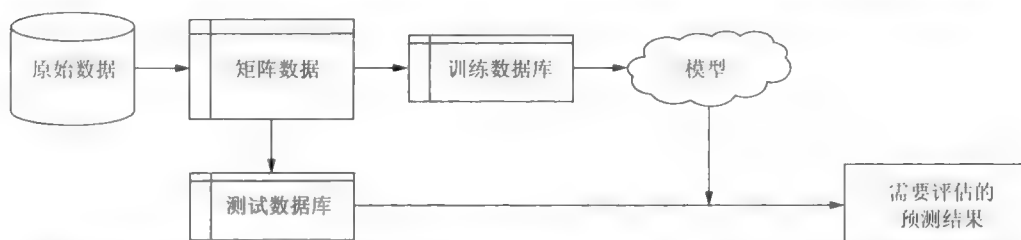
与信赖再带入误差相比，更好的方式是评估模型对其从未见过数据的性能。在前面的章节中我们使用过这种方法，当时我们将数据分成了训练集和测试集。但是在某些情况下，创建训练集和测试集的方式并不总是有用。例如，当数据集很小时，通过划分训练集和测试集来减小样本量是不合适的。

幸运的是，存在一些其他方式来评估模型对其未见过数据的性能。之前我们用来计算性能度量值的 `caret` 添加包也提供了一些函数来实现这个目的。如果想运行 R 代码的示例

但还没有安装该包，请先安装该添加包。还需要使用 `library(caret)` 命令在 R 中加载这个添加包。

10.2.1 保持法

在前面章节中，我们将数据划分成训练集和测试集的过程称为**保持法**。如下图所示，训练数据集用来生成模型，然后应用到测试数据集从而生成预测结果进行评估。比较典型的做法是，大约 1/3 的数据用来测试，2/3 的数据用来训练模型，但是这个比例是可以根据数据量不断变化的。为了确保训练数据和测试数据没有系统性偏差，样本被随机地分成两组。



因为保持法对未来的性能进行真实的精确估计，所以绝不允许测试数据集的结果影响模型。如果基于重复测试的结果选择一个最好的模型，那么很容易在不知不觉中就违反了这个原则。一种替代方法是继续对数据进行划分，在训练数据集和测试数据集之外，分出第三个数据集，也就是验证数据集。验证数据集用来对模型进行迭代和改善，测试数据集只使用一次，在最后的步骤中输出对未来预测的错误率的估计。一种典型的划分方式是 50% 的训练数据集、25% 的测试数据集和 25% 的验证数据集。



细心的读者可能会注意到这种保持测试数据的方法在前面的章节中用来比较不同的模型。这样做实际上违反了之前介绍的原则，因此那些测试数据如果更精确地说应该称为验证数据。如果我们使用测试数据来做决策，我们使用摘樱桃的法则从结果中挑选最好的模型，那么评估将不再是对未来性能的无偏估计。

创建保持样本的一种简单方式是使用随机数发生器将记录划分到各个数据集。这种技术最早在第 5 章用来创建训练数据集和测试数据集。



如果你想执行后面的例子，那么你需要到 Packt 出版社网站上下载 `credit.csv` 个数据集，然后使用命令 `credit <- read.csv("credit.csv")` 来加载数据集。

假设我们有一个名为 `credit` 的 1000 行的数据集，我们可以将其分为 3 部分：

```

> random_ids <- order(runif(1000))
> credit_train <- credit[random_ids[1:500],]
> credit_validate <- credit[random_ids[501:750], ]
  
```

```
> credit_test <- credit[random_ids[751:1000], ]
```

第一行代码将从 1 ~ 1000 的行 ID 随机排序，然后用这些 ID 将 `credit` 数据框分成 500、250 和 250 条记录的 3 个子集，分别对应训练集、验证集和测试集。

这种保持抽样的方式存在一个问题，每个划分包含不同类别的数量可能过大或者过小。在某些特定的情况下，尤其是某些类别本来比例就很小时，这可能导致训练集中不包含该类数据的问题——这是个非常重要的问题，因为模型将无法学习该类别。

为了降低这种情况发生的可能性，我们可以使用**分层随机抽样**的方法。虽然在平均的情况下，简单随机抽样包含的各类别的比例大概与总体数据集中的比例相同，但是分层随机抽样可以确保随机划分后每个类别的比例与总体数据中的比例近似相等。

`caret` 添加包提供了 `createDataPartition()` 函数，它可以基于分层抽样方法来创建随机的划分。对 `credit` 数据集创建训练集和测试集的分层样本的代码如下所示。使用该函数时，需要指定类别向量（这里，`default` 变量表示是否贷款违约），此外，参数 `p` 表示包含在该划分中样本的比例。参数 `list = FALSE` 防止结果存储成列表的形式。

```
> in_train <- createDataPartition(credit$default, p = 0.75,
  list = FALSE)
> credit_train <- credit[in_train, ]
> credit_test <- credit[-in_train, ]
```

向量 `in_train` 表示包含在训练样本中的行号。我们可以使用这些行号为数据框 `credit_train` 选择样本。类似地，通过使用负号，我们通过不包含在 `in_train` 向量中的行号来得到 `credit_test` 数据集。



因为模型如果在更大的数据集中进行训练可以得到更好的性能，所以常见的做法是在选择和评估了最终的模型之后，将模型在整个数据集（训练集加上验证集加上测试集）上重新训练，使得模型能够最大化地利用所有的数据。

虽然这种方式可以确保分布的均匀性，但是分层抽样并不能保证其他类型的代表性。有些样本包含过多或者过少的困难样本、易预测样本或者极端值。尤其是当数据量少时特别明显，每个部分的数据集不足以包含所有的情况。

除了可能的有偏样本以外，保持法的另一个问题是大量的数据部分都被用来测试和验证模型。在模型的性能被度量之前都无法用来训练模型。这种性能估计的方式过于保守。

一种称为**重复保持**的技术有时用来缓解随机构建训练集的问题。重复保持法是保持法的一种特殊形式，它对多个随机保持样本的模型分别评估，然后用结果的均值来评价整个模型的性能。使用多重保持样本时，模型在无代表性数据上训练和测试的可能性就比较小了。我们将在下一节中对这个思路进行扩展。

10.2.2 交叉验证

重复保持法是 **k 折交叉验证**（或者 **k 折 CV**）的基础。**k 折交叉验证**已经成为业界评估模型性能的标准。与可能对同一条记录使用多次重复随机抽样的方法不同，**k 折 CV** 将数据随

机地分成 k 个完全分隔开的部分，这些部分称为折。

虽然 k 可以设置为任意的数值，但是到目前为止，最常用的惯例是使用 10 折交叉验证（10 折 CV）。为什么是 10 折？经验证据告诉我们，使用更大的数后带来的好处并不明显。对于这 10 折中的每一折（每折包含总数据中的 10%），机器学习模型使用剩下 90% 的数据建模。包含 10% 数据的这一折用来评估。训练和评估模型的过程重复 10 次（10 次不同的训练和测试）之后，将输出所有折的平均性能指标。



k 折 CV 的一个极端情况是留一交叉验证法，它将每个样本作为 1 折，确保可以最大数目的样本来建模。虽然看上去很有用，但是计算的代价过大，因此在实践中很少使用。

我们可以使用 `caret` 添加包中的 `createFolds()` 函数来创建交叉验证的数据集。与分层随机保持抽样类似，该函数也尝试在每一折中维持与原始数据类似的各类别的比例。下面的命令可以创建 10 折：

```
> folds <- createFolds(credit$default, k = 10)
```

`createFolds()` 的结果是一个列表，包含了 10 个向量，每个向量都是这一折所抽取数据的行号。我们使用 `str()` 函数来查看其中的内容：

```
> str(folds)
List of 10
 $ Fold01: int [1:100] 1 5 12 13 19 21 25 32 36 38 ...
 $ Fold02: int [1:100] 16 49 78 81 84 93 105 108 128 134 ...
 $ Fold03: int [1:100] 15 48 60 67 76 91 102 109 117 123 ...
 $ Fold04: int [1:100] 24 28 59 64 75 85 95 97 99 104 ...
 $ Fold05: int [1:100] 9 10 23 27 29 34 37 39 53 61 ...
 $ Fold06: int [1:100] 4 8 41 55 58 103 118 121 144 146 ...
 $ Fold07: int [1:100] 2 3 7 11 14 33 40 45 51 57 ...
 $ Fold08: int [1:100] 17 30 35 52 70 107 113 129 133 137 ...
 $ Fold09: int [1:100] 6 20 26 31 42 44 46 63 79 101 ...
 $ Fold10: int [1:100] 18 22 43 50 68 77 80 88 106 111 ...
```

我们可以看到第一折的名称是 `Fold01`，包含了 100 个整数用于表示第一折的数据在 `credit` 数据框中的行号。要创建训练集和测试集来建模和评估，需要一个额外的步骤。下面的代码显示了通过第一折创建数据的过程。与我们在分层抽样时所做的一样，我们用选出来的样本当作训练集，用负号把剩余的样本放到测试集：

```
> credit01_train <- credit[folds$Fold01, ]
> credit01_test <- credit[-folds$Fold01, ]
```

要想执行完整的 10 折 CV，这个步骤需要重复总共 10 次，每一次都要建立模型，然后计算模型的性能。最后，通过对所有的性能度量取平均值得到总体的性能。幸运的是，我们可以使用之前学过的一些技术来自动完成这个过程。

为了演示这个过程，我们用 10 折 CV 方法对 `credit` 数据集建立 C5.0 决策树模型，然后估计 Kappa 统计量。首先，我们需要加载 `caret` 添加包（创建折）、`C50` 添加包（决策树

模型) 和 `irr` 添加包 (计算 Kappa 值)。选择后面的两个添加包主要是为了演示这个例子。如果愿意, 也可以使用不同的模型或者计算不同的统计量来完整这个过程。

```
> library(caret)
> library(C50)
> library(irr)
```

其次, 我们像先前做的那样创建 10 折的列表, 使用 `set.seed()` 函数是为了保证读者自己运行后面的代码的结果和本书中的一致:

```
> set.seed(123)
> folds <- createFolds(credit$default, k = 10)
```

最后, 使用 `lapply()` 对列表的每个元素进行相同的操作。如以下示例代码所示, 因为没有现成的函数可以完成我们的需求, 所以我们需要定义自己的函数来传给 `lapply()`。我们定制的函数可以将 `credit` 数据框分成训练集和测试集, 使用 `C5.0()` 函数对训练集的数据建立决策树模型, 然后对测试数据进行预测, 并使用 `kappa2()` 函数来比较预测值和真实值:

```
> cv_results <- lapply(folds, function(x) {
  credit_train <- credit[x, ]
  credit_test <- credit[-x, ]
  credit_model <- C5.0(default ~ ., data = credit_train)
  credit_pred <- predict(credit_model, credit_test)
  credit_actual <- credit_test$default
  kappa <- kappa2(data.frame(credit_actual, credit_pred))$value
  return(kappa)
})
```

结果的 Kappa 统计量保存在 `cv_results` 对象的列表中, 我们可以使用 `str()` 函数进行查看:

```
> str(cv_results)
List of 10
 $ Fold01: num 0.283
 $ Fold02: num 0.108
 $ Fold03: num 0.326
 $ Fold04: num 0.162
 $ Fold05: num 0.243
 $ Fold06: num 0.257
 $ Fold07: num 0.0355
 $ Fold08: num 0.0761
 $ Fold09: num 0.241
 $ Fold10: num 0.253
```

这样, 我们把 10 折 ID 列表转成了 kappa 统计量的列表。只剩下最后一个步骤: 计算这 10 个数的平均值。你可能会尝试输入 `mean(cv_results)`, 但由于 `cv_results` 并不是数值向量, 所以会报错。实际上, 应该使用 `unlist()` 函数, 它可以消除列表的结构, 将 `cv_results` 简化成一个数值向量, 然后我们可以像期望的那样计算 Kappa 值:

```
> mean(unlist(cv_results))
```


[1] 0.1984929

不幸的是，这个 κ 值非常低（事实上，在解释评分体系中对应着“很差”，说明这个信用记分模型的效果并不比随机猜测好很多。在下一章中，我们将基于 10 折 CV 方法研究自动方法来帮助我们改进这个模型的性能。



也许当前能可靠地估计模型性能的最好标准方法是重复 k 折 CV。从字面意思也可以猜测出，它重复地应用 k 折 CV 方法，然后对结果求平均值。常用的策略是将 10 折 CV 执行 10 次。虽然会增加运算的复杂度，但是可以得到稳健的估计。

10.2.3 自助法抽样

还有一种虽然不如 k 折 CV 这么受欢迎但是也被广泛使用的方法是自助法抽样（bootstrap sampling），简称自助法（bootstrap）。一般来说，它主要指一些统计方法，通过对数据进行随机抽样的方式来估计大数据集的内容。当该原理应用到机器学习模型性能时，意味着创建一些随机选取的训练集和测试集，然后用来估计性能的统计量。各种随机产生的数据集的结果可以通过平均值计算得到一个最终的估计值，用来评估未来的性能。

那么，这个过程与 k 折 CV 有什么不同呢？交叉验证方式将数据分成彼此分隔的部分，每个样本只能出现一次，而自助法通过有放回的抽样方式使得每个样本可以被选择多次。这意味着假设原始数据包含 n 个样本，那么自助法可以创建一个或者多个仍然包含 n 个样本的数据集，有些样本是重复的。相应的测试数据集仍然由未选入训练集中的样本来构建。

使用之前提到的有放回的抽样方式，每个样本包含在训练集中的概率是 63.2%。因此，样本包含在测试集中的概率是 36.8%。换句话说，测试集只能代表 63.2% 的可能样本，因为很多样本重复了。相比之下，10 折 CV 方法可以使用 90% 的数据用来训练，自助法抽样对完整数据集的代表性更弱。

显然，只利用 63.2% 的数据训练出来的模型的性能不如更大量数据训练出来的模型，对自助法性能的估计也远不如使用全体数据训练的结果。有一种特殊的自助法的情况可以处理这个问题，称为 0.632 自助法，通过训练数据集（过于乐观）和测试数据集（过于悲观）的函数来计算最终的性能度量。最终的错误率可以由以下公式计算：

$$\text{错误率} = 0.632 \times \text{错误率}_{\text{测试}} + 0.368 \times \text{错误率}_{\text{训练}}$$

自助法比交叉验证具有一个优势，它对于小数据集的效果更好。此外，自助法抽样在性能度量之外还有其他的应用。特别地，第 11 章将学习如何利用自助法抽样的原理来改善模型的性能。

10.3 总结

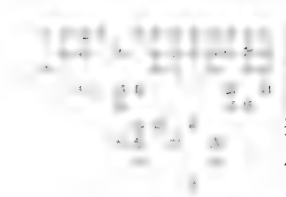
本章介绍了评估机器学习分类模型性能最常用的一些度量方法和技术。虽然准确度提供

了一种简单的方式来考察模型是如何正确，但是在一些不常见事件的情况下这种方法会产生误导性，因为真实世界中这类事件的成本常常与它们在数据中出现的频率成反比。

基于混淆矩阵的一些度量方式可以在各种类型的误差成本之间更好地捕捉到一个平衡点。在灵敏度与特异性之间或者精确度与回溯精确度之间进行权衡是一个很有用的工具，可以帮助考虑真实世界中误差的含义。ROC 曲线等可视化方法也有助于这个目的。

同样值得注意的是，有时候对于模型性能的最好度量需要考虑其是否满足或者不满足一些其他的目标。例如，你可能需要能够用简单的语言来解释模型的逻辑，那么很多模型就会不在考虑之内。此外，即使模型的性能非常好，但如果计算速度太慢或者很难扩展到生产环境中，那么它也是无用的。

对于度量性能还有一个很明显的扩展，就是确定自动方式来针对特定的任务找到最好的模型。在第 11 章中，我们将基于目前为止的工作，研究通过系统地迭代、改善以及合并多种算法来建立更智能模型的方式。



提高模型的性能

当一个运动队无法达到他们的目标时（得到奥运金牌、联赛冠军或者创世界纪录），它需要开始一个新的过程来寻求进步的空间，从而避免将来相似的命运。假设你是该队的教练，你会如何安排训练的内容呢？你可能会让运动员训练得更加刻苦或者改变训练方式来开发他们所有的潜能。或者，可以更加强调团队协作，对每位队员的优势和不足进行更好地利用。

现在假设你是一个以寻求世界冠军机器学习算法为任务的教练——可以是参加 Kaggle 网站（<http://www.kaggle.com/competitions>）竞赛，去赢取百万 Netflix Prize（<http://www.netflixprize.com>），或者只是简单地提高生意的业绩。你会如何开始？

虽然竞赛的内容可能不同，但是很多能够提高运动队成绩的策略也可以用来提高统计学习器的性能。作为一个教练，工作就是寻找训练技术和团队协作技巧的组合，从而实现成绩上的目标。

本章将基于之前的内容介绍一些能提高机器学习算法预测性能的技术，你会学到：

- 如何通过寻找训练条件的优化集合来调整机器学习模型的性能。
- 将多个模型组合起来从而处理更富挑战性问题的方法。
- 获得机器学习算法最大程度性能的最先进的技术。

并不是所有这些方法都适用于每个问题，但是你会发现机器学习竞赛中的胜利者至少会用到其中一种方法。为了保持竞争性，你也需要将这些方法加入到你的技能集。

11.1 调整多个模型来提高性能

有些机器学习问题都很适合使用前几章介绍的多个模型。在这种情况下，我们不需要花太多的时间进行迭代和优化，它已经足够好了。另一方面，有些问题本质上就很难。需要学

习的潜在概念极其复杂，需要对很多微妙的关系具有很好的理解，或者可能具有随机元素，使得从噪声中分离有用的信号变得很难。

对这些如此困难的问题开发性能极好的模型是一门科学，同时也是一门艺术。有时候，找出可以提高性能的领域是需要一些直觉的。而另一些情况下，实现性能提高可能需要使用蛮力和反复试验的方式。当然，寻找各种可能的提高方式的过程可以通过使用自动化的程序来辅助。

在第 5 章中，我们尝试了一个复杂的问题，识别可能违约的贷款。虽然我们可以使用性能调节方法得到一个可以接受的分类准确度 72%，但是根据第 10 章更仔细地审视后，我们认识到这种高准确度可能会造成误导。尽管准确度非常合理，但是 Kappa 统计量只有大约 0.2，这说明了该模型实际的性能其实是有些糟糕的。在这一节中，我们将重新研究这个信用评分模型，看看如何对结果进行改善。



要是想跟着例子运行程序，需要到 Packt 出版社网站下载 `credit.csv` 文件，然后将它保存到 R 的工作目录中。使用命令 `credit <- read.csv("credit.csv")` 将文件加载到数据框。

你可能还记得，我们最首先使用股票 C5.0 决策树来对这个信用数据建立分类模型。然后我们尝试通过调节 `trials` 参数来增加提升（boosting）迭代的次数从而提升模型的性能。通过将默认的 1 次增加到 10 和 100 次，可以增加模型的准确度。调节模型合适的选项的过程称为**参数调整**。

参数调整并不仅限于决策树。例如，我们通过寻找最合适的 k 值来调整 k 近邻模型、在神经网络和支持向量机模型中使用大量的选项（比如，调节节点、隐层的数目，或者选择不同的核函数）。大多数机器学习算法都可以调整至少一个参数，而大多数复杂的模型都提供了很大数目的方式来调整模型从而进行更好的拟合。虽然这可以让模型更适合数据，但是尝试所有可能选项的复杂度是很吓人的。需要使用一种更系统的方式。

使用 **caret** 进行自动参数调整

与其对模型的每个参数选择任意值，（这样做不仅耗时而且不科学），不如对搜索参数值的过程进行引导从而找到最优的组合。

我们第 10 章广泛使用的 `caret` 添加包提供了很多工具可以帮助自动参数调整。最核心的功能是提供了一个 `train()` 函数作为标准接口，为分类和回归任务训练 150 种不同的机器学习模型。使用该函数时，可以选择评估的方法和度量，它自动搜寻一个最优的模型。



不要被这么多个模型吓到了——我们在之前的章节中已经见到了很多。其他的很多模型都是这些基本概念的扩展或者变种。在当前学习的基础上，你要相信你已经具备了理解这 150 种选择的能力。

自动参数调整需要考虑以下 3 个问题：

□ 需要使用数据来训练哪种机器学习模型（或者算法的具体实现）？

□ 哪些模型参数是可以调整的，它们能调整的空间有多大？

□ 使用何种评价标准来评估模型从而找到最优的候选者？

要回答第一个问题需要在机器学习的任务和 150 个模型之间做适当的匹配。显然，需要对这些机器学习模型的广度和深度有很好的理解。本书提供了前者所需要的背景，额外的练习对后者很有用。此外，排除法也很有用：通过判断任务是分类还是回归就可以排除几乎一半的模型；其他的可以通过数据的形式或者避免黑箱的模型来排除。不管怎样，你还可以尝试多种模型，然后通过比较模型的结果来选择一个最好的。

第二个问题很大程度上是被模型的选择所决定的，因为每个算法使用唯一的一套参数。本书涉及的预测模型的可调节参数都列在下表中。注意，虽然有些模型还有没列出来的选项，但是 caret 添加包只支持下表中列出的选项。



要想知道这 150 个模型以及相关的可调节参数的详细信息，可以参考 caret 添加包的作者 Max Kuhn 提供的表：<http://caret.r-forge.r-project.org/modelList.html>。

模型	学习任务	方法名	参数
k 近邻	分类	knn	k
朴素贝叶斯	分类	nb	fL, usekernel
决策树	分类	C5.0	model, trials, winnow
OneR 规则学习器	分类	OneR	无
RIPPER 规则学习器	分类	JRip	NumOpt
线性回归	回归	lm	无
回归树	回归	rpart	cp
模型树	回归	M5	pruned, smoothed, rules
神经网络	二者皆可	nnet	size, decay
支持向量机（线性核）	二者皆可	svmLinear	C
支持向量机（径向基核）	二者皆可	svmRadial	C, sigma
随机森林	二者皆可	rf	mtry

自动调整的目标是搜索候选模型的集合，该集合由所有参数组合的矩阵或者网格的形式构成。因为要想遍历所有参数的所有可能值是难以实现的，所以只选择参数可能值的一些子集来构建网格。caret 默认对每个参数最多搜索 3 个可能值，假设一共有 p 个参数，这意味着 3^p 个候选模型将会被测试。例如，默认情况下，自动调整的 k 近邻模型将比较 $3^1=3$ 个候选模型，比方说 $k=5$ 、 $k=7$ 和 $k=9$ 。类似地，调整决策树模型将比较 27 个不同的候选模型，由 model、trials 和 winnow 值组成的 $3^3=27$ 种可能的网格来实现。在实践中，只有 12 个模型被实际测试到，因为 model 和 winnow 参数只能选取两个值（分别是 tree 与 rules 和 TRUE 与 FALSE），那么网格的大小是 $3 \times 2 \times 2=12$ 。



因为 `caret` 默认的搜索网格对于实际的机器学习问题可能不是很理想，所以该函数还允许用户自定义搜索网格，通过简单的命令即可定义。我们会在后面进行介绍。

自动模型调整的第三个也是最后一个步骤是选择一种方法从候选者中识别最好的模型。我们使用第 10 章讨论过的方法（比如，选择重采样的策略）来创建训练集和测试集，或者使用度量预测准确度的模型性能统计量。

`Caret` 添加包支持我们介绍过的所有重采样策略和大部分的性能统计量，包括准确度和 Kappa 值（用于分类器）以及 R 方值或者 RMSE（用于数值模型）等统计量。如果需要的话，也可以使用代价敏感度方面的度量方式，比如灵敏度、特异性、ROC 曲线下面积（AUC）等。

默认情况下，`caret` 在选择最优模型时，通过这些性能度量指标的最大值来选择。因为这样的方法在实践中有时会通过大量增加模型复杂度的方式来选择边际性能递增的模型，该添加包也提供了可供选择的其他函数。

在各种选项中，大部分默认值都是比较合理的。例如，在分类模型中使用自助法抽样的预测准确度来选择最优性能的优化器。从这些默认值开始，我们使用 `train()` 函数来设计各种实验。

1. 创建简单的调整模型

为了说明调整模型的过程，先来看看当我们尝试使用 `caret` 添加包的默认设置来调整信用评分模型时会发生什么。从这里我们可以学到如何调整选项。

最简单的调整模型的方式只需要通过 `method` 参数来指定模型的类型。因为我们之前使用 C5.0 决策树的方法对信用数据建模，所以我们通过优化模型的方式来继续之前的工作。使用默认设置来调整 C5.0 决策树的基础 `train()` 的命令如下：

```
> library(caret)
> set.seed(300)
> m <- train(default ~ ., data = credit, method = "C5.0")
```

首先，`set.seed()` 函数用来初始化 R 的随机数发生器。我们在前面的章节里用过很多次。通过设定 `seed` 参数（这里我们设置为一个任意的 300），可以使随机数遵循一个预先设定的序列。这可以使得类似 `train()` 这样使用随机抽样的模拟方法能够在重复运行中得到相同的结果——如果分享代码并尝试得到之前结果，那么这是非常有用的。

其次，R 的公式接口可以把树定义成 `default~.`。它表示我们对贷款违约的状态（yes 或者 no）进行建模，使用了信用数据集中的所有其他变量。参数 `method = "C5.0"` 告诉 `caret` 使用 C5.0 决策树算法。

输入了之前的命令后，开始执行调整过程，可能会有显著的延迟（取决于计算机的配置）。即使是很少量的数据集，也需要非常巨大的计算量。R 会重复地生成数据的随机抽样，建立决策树模型，计算性能统计量，并且对结果进行评估。

实验的结果存储到一个对象中，我们命名为 `m`。如果要查看该对象的内容，命令 `str(m)` 会列出所有相关的数据——但是这种方式得到的信息太多了。简单地键入该变量

名，就会得到摘要的结果。例如，键入 `m` 后会产生如下的结果：

```

1 1000 samples
   16 predictors
   2 classes: 'no', 'yes'

2 No pre-processing
  Resampling: Bootstrap (25 reps)

  Summary of sample sizes: 1000, 1000, 1000, 1000, 1000, 1000, ...

3 Resampling results across tuning parameters:

  model trials winnow Accuracy Kappa Accuracy SD Kappa SD
  rules 1      FALSE  0.685   0.258  0.0256     0.0562
  rules 1      TRUE   0.689   0.255  0.0268     0.057
  rules 10     FALSE  0.711   0.309  0.0209     0.0459
  rules 10     TRUE   0.711   0.304  0.0195     0.0448
  rules 20     FALSE  0.722   0.326  0.0198     0.0451
  rules 20     TRUE   0.723   0.327  0.0184     0.0371
  tree  1      FALSE  0.677   0.229  0.0303     0.07
  tree  1      TRUE   0.677   0.222  0.027     0.0596
  tree  10     FALSE  0.722   0.288  0.0206     0.056
  tree  10     TRUE   0.717   0.278  0.017     0.0436
  tree  20     FALSE  0.73    0.307  0.0201     0.0562
  tree  20     TRUE   0.729   0.306  0.015     0.0415

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were model = tree, trials = 20
and winnow = FALSE.

```

该结果包含以下几个主要的部分：

1) **输入数据的简单描述**：如果你熟悉数据并且正确地应用了 `train()` 函数，那么在该信息中不会看到意想不到的结果。

2) **预处理和重抽样方法应用情况的信息**：我们可以看到 25 个自助法样本，每一个都包含 1000 个样本，用来建模。

3) **候选模型评估的列表**：在这一节里，我们可以确信 12 个不同的模型被测试到了，基于 3 个 C5.0 调整参数的组合：`model`、`trials` 和 `winnow`。每个候选模型的准确度和 Kappa 统计量的均值和标准差（标记为 SD）都得到了展示。

4) **最佳模型的选择**：根据提示，具有最大准确度（0.73）的模型被选为最佳。该模型使用的参数情况为 `model = tree`、`trials = 20`、`winnow = FALSE`。

`train()` 函数使用最佳模型（前面 4，介绍的）中的参数对所有数据建立模型，并将其存储在 `m$finalModel` 对象中。在大多数情况下，无需直接操作 `finalModel` 子对象。而是使用 `predict()` 函数通过 `m` 对象来进行预测，同时还提供了一些额外的功能，我们马上会进行介绍。例如，将最佳模型应用到训练数据中进行预测，可以使用以下命令：

```
> p <- predict(m, credit)
```

预测的结果向量可以像我们之前做过很多次的那样操作：

```
> table(p, credit$default)
```

```

p      no yes
no    700  2
yes     0 298

```

在用来训练最终模型的 1000 个样本中，只有 2 个被错误地分类。记住，这只是重新代入误差，不能看成是对未来数据性能的度量。自助法估计的 73%（输出结果中有显示）是对

未来性能的一个更现实的估计。

之前提到过，将 `predict()` 直接用于 `train()` 对象还会有额外的好处，注意不是调用 `finalModel` 子对象或者使用最佳模型的参数来训练新模型。

首先，`train()` 函数应用到数据中的任何数据预处理方法也会用类似的方式应用到产生预测的数据中。这包括中心化和标准化（使用 `k` 近邻）的数据转换、缺失值处理以及一些其他方法。这确保用来建模的数据准备步骤在模型部署后仍然保留。

其次，`predict()` 函数提供了标准的接口用来得到预测的类别值和概率——即使是通常的模型也需要额外的步骤来得到这些信息。预测的类别会默认提供，如下所示：

```
> head(predict(m, credit))
[1] no yes no no yes no
Levels: no yes
```

要想得到每一类估计的概率，只需要设置一个额外的参数 `type = "prob"`：

```
> head(predict(m, credit, type = "prob"))
      no      yes
1 0.9606970 0.03930299
2 0.1388444 0.86115561
3 1.0000000 0.00000000
4 0.7720279 0.22797208
5 0.2948062 0.70519385
6 0.8583715 0.14162851
```

即使有些情况下底层的模型会使用不同的字符串来表示预测概率值（比如，朴素贝叶斯模型使用 "raw"），但 `caret` 自动在后台将 `type = "prob"` 转成其需要的形式。

2. 定制调整的过程

我们之前创建的决策树证明 `caret` 添加包可以在最少介入下生成最优模型。默认设置可以使高性能的模型能够很容易创建。但是，如果不进行深入研究，你可能会错过性能方面最高端的部分。或者你可能需要改变默认的评估标准从而更适合自己的机器学习问题。之前介绍的每一个步骤都是可以针对具体学习任务进行定制。

为了更灵活地说明这些，我们对之前介绍的信贷决策树的工作进行一些修改，从而反映我们在第 10 章使用的过程。如果你还记得这一章的话，我们使用了 10 折交叉验证来估计 Kappa 统计量。这里我们会做相同的事情，使用 Kappa 优化决策树的提升（boosting）参数（提升第 5 章介绍的决策树的准确度）。

`trainControl()` 函数用来创建一系列的配置选项，也称为控制对象，与 `train()` 函数一起使用。这些选项考虑到了诸如重抽样策略以及用于选择最佳模型的度量这些模型评价标准的管理。虽然该函数可以用于几乎所有参数调整的方面，但是我们只专注于两个重要的参数：`method` 和 `selectionFunction`。



如果你期望了解得更详细，可以使用 `?trainControl` 帮助命令来得到所有参数的列表。

对于 `trainControl()` 函数, `method` 参数用来设置重抽样的方法, 例如保持抽样或者 k 折交叉验证。下表列出了 `caret` 调用这些方法时使用的缩写, 以及用来调节样本量和迭代次数的所有额外参数。虽然这些重抽样方法的默认值遵循最受欢迎的惯例, 但你也可以根据自己的样本量和模型的复杂度进行调整。

重抽样方法	方法名	额外的选项和默认值
保持抽样	LGOCV	<code>p = 0.75</code> (训练数据比例)
k 折交叉验证	<code>cv</code>	<code>number = 10</code> (折的数目)
重复 k 折交叉验证	<code>repeatedcv</code>	<code>number = 10</code> (折的数目) <code>repeats = 10</code> (迭代的次数)
自助法抽样	<code>boot</code>	<code>number = 25</code> (重抽样迭代的次数)
0.632 自助法	<code>boot632</code>	<code>number = 25</code> (重抽样迭代的次数)
留一交叉验证法	LOOCV	无

`trainControl()` 函数的 `selectionFunction` 参数可以设定一函数用来在各个候选者中选择最优的模型。其中包含了 3 个函数。`best` 函数简单地选择具有最好的某特定度量值的候选者, 这是默认的选项。另外两个函数用来在最好模型性能的特定阈值之内选择最节俭的 (或者说最简单的) 模型。`oneSE` 函数选择最好性能标准差之内的最简单的候选者。`Tolerance` 选择某个用户指定比例之内的最简单的候选者。



`caret` 添加包使用简易度来对模型进行排序容易陷入主观。要想知道模型是如何排序的, 可以通过使用 `?best` 帮助函数来得到帮助页面。

使用 10 折交叉验证和 `oneSE` 选择函数可以创建一个名为 `ctrl` 的控制对象, 如下代码所示。注意 `number = 10` 只是为了演示得更清楚, 因为它是 `method = "cv"` 时的默认值, 本来是可以省略的。

```
> ctrl <- trainControl(method = "cv", number = 10,
  selectionFunction = "oneSE")
```

我们可以马上使用该函数的结果。

同时, 我们定义我们实验的下一个步骤是创建用来优化参数的网格。网格的每一列表示模型中的一个参数, 前面用点号作为前缀。因为使用 C5.0 决策树, 所以这意味着我们名为 `.model`、`.trials` 和 `.winnow` 的列。对于其他模型, 参考本章之前的表。数据框中的每一行代表一种特定的参数值的组合。

如果不想自己手工创建这样的数据框 (如果参数值的可能组合数太多时就非常困难), 可以使用 `expand.grid()` 函数, 它能利用所有值的组合创建数据框。例如, 假设需要参数 `model = "tree"` 和 `winnow = "FALSE"` 保持不变, 同时 `trials` 需要选择 8 个值。可以使用如下方式创建:

```
> grid <- expand.grid(.model = "tree",
  .trials = c(1, 5, 10, 15, 20, 25, 30, 35),
  .winnow = "FALSE")
```

结果数据框 grid 包含 $1 \times 8 \times 1 = 8$ 行:

```
> grid
  .model .trials .winnow
1  tree      1  FALSE
2  tree      5  FALSE
3  tree     10  FALSE
4  tree     15  FALSE
5  tree     20  FALSE
6  tree     25  FALSE
7  tree     30  FALSE
8  tree     35  FALSE
```

每一行都可以用来创建一个候选模型, 使用该行中模型参数的组合来构建。

给定这个搜索网格和之前创建的控制列表后, 我们已经准备好运行完全定制的 `train()` 实验。与之前一样, 我们设置随机数种子来确保可重复的结果。但是这一次, 我们传送的是自己定制的控制对象和调整参数的网格, 同时添加参数 `metric = "Kappa"`, 表示模型评估函数用到的统计量——在这个例子中, 是 `oneSE`。完整的命令如下所示:

```
> set.seed(300)
> m <- train(default ~ ., data = credit, method = "C5.0",
             metric = "Kappa",
             trControl = ctrl,
             tuneGrid = grid)
```

结果是一个对象, 我们可以像之前一样来查看:

```
> m

1000 samples
 16 predictors
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-validation (10 fold)

Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...

Resampling results across tuning parameters:

  trials Accuracy  Kappa Accuracy SD  Kappa SD
1       0.724     0.312 0.0255     0.059
5       0.713     0.292 0.0211     0.0602
10      0.719     0.295 0.0311     0.0672
15      0.721     0.301 0.0197     0.0511
20      0.717     0.293 0.0279     0.0791
25      0.728     0.315 0.0322     0.0937
30      0.729     0.31  0.0277     0.0807
35      0.741     0.339 0.0314     0.0935

Tuning parameter 'model' was held constant at a value of 'tree'
Tuning parameter 'winnow' was held constant at a value of 'FALSE'
Kappa was used to select the optimal model using the one SE rule.
The final values used for the model were model = tree, trials = 1
and winnow = FALSE.
```

虽然很多结果都与之前调整模型的一样, 但是也有一些不同。因为使用了 10 折交叉验证, 所以建立候选模型的样本量减到了 900, 而不是自助法中的 1000。按照我们的要求, 对 8 个候选模型进行了测试。此外, 因为 `model` 和 `winnow` 保持不变, 所以它们的值不再显示在结果中, 而是显示在脚注中。

这里的最佳模型与先前的实验有很大的不同。之前，最佳模型使用了 `trials = 20`，但是这里最佳模型使用了 `trials = 1`。这看起来很奇怪，实际上因为我们使用了 `oneSE` 规则而不是之前的 `best` 规则来选择模型。即使 `trials` 为 35 的模型按照 `Kappa` 值提供了最好的性能，但是 `trials` 为 1 的模型也具有差不多的性能，但是要简单得多。简单的模型更可取，不仅是因为它们具有更好的计算性能，而且还能减少过拟合的可能性。

11.2 使用元学习来提高模型的性能

另一种提高单个模型性能的方法是将多个模型合并成一个更强的组。正如最好的运动队通常都是拥有能力互补的队员，而不是能力互相重叠，最好的机器学习算法也会利用多个互补的模型的组合。因为模型对特定学习任务具有独特的偏爱，所以它可能对样本的某个子集很适合但是对其他的不适合。因此，通过使用几个不同组员的能力，可以创建组员较弱但整个小组很强的模型。

这种组合和管理多个模型的预测技术属于一套更广泛的元学习方法，该方法包含了所有涉及如何学习的技术。它可以包含从通过自动迭代设计决策来提升性能的简单算法（例如，使用本章前面提到的自动调整参数的方法），到借鉴了进化生物学和遗传学的自修改和自适应学习方式的复杂算法。

在本章的剩余部分，我们将关注元学习，只因为它适合对多个模型的预测和要求的结果之间的关系建模。这里包含的团队学习技术非常强大，也经常用来建立更有效的分类器。

11.2.1 理解集成学习

假设你是一个电视问答节目的参与者，正在回答百万大奖的最后一个问题，可以选择由 5 位好友组成的亲友团帮忙答题。大多数人会选择各个不同领域的专家。例如，该亲友团包括文学、科学、历史、艺术方面的教授，以及一个熟悉当前流行文化的专家，这是一个稳妥而全面的小组。因为他们的知识面很广，所以不太可能遇到他们都不熟悉的问题。

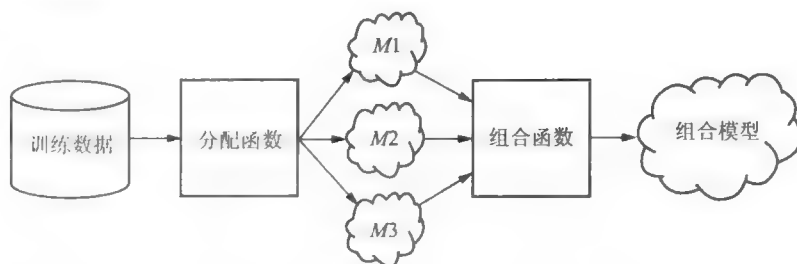
利用类似的创建一个多样性专家组的原理的元学习方法也称为集成学习（ensemble）。所有的集成学习方法都是基于结合多个很弱的学习器来创建一个很强学习器的思路。用这个简单的原理，可以开发各种算法，通过以下两个问题来区分：

- 如何选择或者构造那些较弱的学习模型？
- 如何将这些较弱的学习模型的预测结果组合起来形成最终的预测？

在回答这些问题时，依据以下的流程图来想象集成的过程是很有帮助的，几乎所有的集成方法都遵循这个模式。

首先，输入训练数据用来建立很多个模型。分配函数决定每个模型接收完整的训练数据集还是某个抽样的样本。因为理想的集成学习包含各种不同的模型，所以分配函数可以通过人工改变输入数据来训练各种模型的方式来增强多样性。例如，它可以使用自助法抽样来构造单一的训练数据集，或者将不同特征的子集或样本传送给每个模型。另一方面，如果集成

学习已经包含了多种算法（例如，神经网络、决策树以及 k 近邻分类器），那么分配函数将传送相对没改变的数据。



创建模型后，它们可以用来产生一系列的预测，这需要用一些方式来管理。**组合函数**用来对预测中的不一致进行调解。例如，集成学习可能利用投票表决来决定最终的预测，或者使用更复杂的策略（例如，根据模型的先验性能）对其票数进行加权。

有些集成学习甚至使用另一个模型从各种预测的组合中学习一个组合函数。例如，当 $M1$ 和 $M2$ 都投票 Yes 而实际常常是 No 时，集成学习可以忽略 $M1$ 和 $M2$ 的投票然后直接预测为 No。这种使用多个模型的预测来训练一个仲裁模型的过程称为**堆叠（stacking）**。

使用集成学习的一个好处是能够节省寻求单一最佳模型的时间。只需要训练一批表现尚可的候选者然后整合它们即可。当然，便利性并不是基于集成学习的方法常常可以在机器学习竞赛中获胜的唯一原因。集成学习与单一模型相比还有很多性能上的优势：

- **对于未来问题更好的普适性：**因为不同学习器的意见都成为了最终预测结果的一部分，所以单一的偏好不会处于主导地位。这可以降低学习时过拟合的可能性。
- **可以提升大量数据或少量数据的性能：**很多模型在处理数目巨大的特征或者样本时常常会遇到内存或者复杂度的限制，训练多个小的数据集比训练单个大的数据集会更有效。此外，集成学习经常使用分布式的计算方法。相反地，集成学习对于最小的数据集也能有很好的表现，因为很多重抽样方法（例如，自助法）本身就是很多集成设计的固有方法。
- **将不同领域数据合成的能力：**因为不存在一刀切的学习算法（可以参考天下没有免费午餐的理论），所以在各种不同领域都在持续产生大数据时，集成学习可以把多种类型的学习器得到的信息整合起来的能力变得越来越重要。
- **对于困难学习任务更细致的理解。**真实世界的想象常常因为各种相互影响错综复杂的因素而非常复杂。将任务分解成很多小部分的模型可以更精确地捕捉到单一的全局模型容易遗漏的细小模式。

如果不能很简单地使用 R 来应用这些方法，那么以上的好处是没有用的，很多 R 添加包要做的正是这些：让我们看看几个最受欢迎的集成学习方法以及它们如何对我们之前的信用模型的性能进行提升。

11.2.2 bagging

得到广泛认可的最好的集成学习方法之一的技术是**自助汇聚法**，简称为 **bagging** 方法。Leo Breiman 在 1994 年对该方法进行过描述，bagging 对原始训练数据使用自助法抽样的方式产生很多个训练数据集。这些数据集使用单一的机器学习算法产生多个模型，然后使用投票（对于分类问题）或者平均（对于数值预测）的方法来组合预测值。



关于 bagging 的更多信息，请参考 Bagging predictors, Machine Learning, Vol. 24, pp. 123-140, by L. Breiman (1996)。

虽然 bagging 是一种相对简单的集成学习器，但是只要它使用相对**不稳定**的学习器就能得到很好的效果，不稳定学习器会随着数据发生的很小变化产生差别很大的模型。不稳定模型是必不可少，因为可以确保当自助法的数据集之间的差异很小时集成学习也能具有很好的多样性。基于这个原因，bagging 经常和决策树一起使用，因为决策树倾向于随着数据的微小变化发生比较大的改变。

ipred 添加包提供了 bagging 决策树的经典实现。bagging() 函数与之前介绍的很多模型的使用方式类似。nbagg 参数控制用来投票的决策树的数目（默认值是 25）。依赖于学习任务的难度和训练数据的数量，增加该数值可以提升模型的性能，一直到某个极限。不利之处是这会带来很多额外的计算量。决策树越多，训练的时间也会越长。

安装 ipred 添加包后，可以按照如下的方式创建集成学习。我们使用默认的 25 个决策树。

```
> library(ipred)
> set.seed(300)
> mybag <- bagging(default ~ ., data = credit, nbagg = 25)
```

产生的模型可以被 predict() 函数使用：

```
> credit_pred <- predict(mybag, credit)
> table(credit_pred, credit$default)
```

```
credit_pred no yes
           no 699  2
           yes  1 298
```

根据之前的结果，该模型看上去对训练数据拟合得非常好。要想知道它在未来的性能方面表现如何，可以通过 caret 添加包中的 train() 函数使用 10 折交叉验证的方法来建立 bagging 树。注意 ipred 添加包中的 bagging 树的函数是 treebag，如下所示：

```
> library(caret)
> set.seed(300)
> ctrl <- trainControl(method = "cv", number = 10)
> train(default ~ ., data = credit, method = "treebag",
        trControl = ctrl)
1000 samples
 16 predictors
  2 classes: 'no', 'yes'
```

```
No pre-processing
Resampling: Cross-Validation (10 fold)

Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...
```

Resampling results

Accuracy	Kappa	Accuracy SD	Kappa SD
0.735	0.33	0.0344	0.0859

Kappa 值是 0.33，说明 bagging 树模型与我们之前通过调整参数得到的最好的 C5.0 决策树模型的效果差不多。

在决策树的 bagging 方法之外，caret 添加包还提供了更通用的 bag() 函数。它对很多模型提供了可以直接使用的支持，而且通过一些额外的努力也可以用于更多类别的模型。bag() 函数使用一个控制对象来配置 bagging 过程。它需要指定 3 个函数：一个用来拟合模型、一个用来进行预测、一个用来聚集投票结果。

例如，要创建一个基于 bagging 的支持向量机 (SVM) 模型，使用第 7 章用过的 kernlab 添加包中的 ksvm() 函数。bag() 函数要求我们提供用来训练 SVM、做预测和统计投票的函数。

我们不需要自己写这些函数，caret 添加包内置的 svmBag 列表对象提供了我们可以使用的这 3 个函数：

```
> str(svmBag)
List of 3
 $ fit      :function (x, y, ...)
 $ pred     :function (object, x)
 $ aggregate: function (x, type = "class")
```

通过查看 svmBag\$fit 函数可以发现，它只是简单地调用了 kernlab 添加包中的 ksvm() 函数：

```
> svmBag$fit
function (x, y, ...)
{
  library(kernlab)
  out <- ksvm(as.matrix(x), y, prob.model = is.factor(y), ...)
  out
}
<environment: namespace:caret>
```

svmBag 中的 pred 和 aggregate 函数也是类似的直接方式。通过学习这几个函数，然后使用相同的格式来创建自己的函数，可以使用 bagging 来实现任意的机器学习算法。



caret 添加包中还包括朴素贝叶斯模型 (nbBag)、决策树 (ctreeBag) 和神经网络 (nnetBag) 的例子。

应用 svmBag 中的这 3 个函数，可以创建一个 bagging 控制对象：

```
> bagctrl <- bagControl(fit = svmBag$fit,
                        predict = svmBag$pred,
                        aggregate = svmBag$aggregate)
```

通过使用我们之前定义的 `train()` 函数和训练控制对象 `ctrl`，可以计算 bagging 的支持向量机模型，如下所示。注意执行这些代码需要 `kernlab` 添加包。如果之前没有安装，就需要先安装它。

```
> set.seed(300)
> svmbag <- train(default ~ ., data = credit, "bag",
                  trControl = ctrl, bagControl = bagctrl)
> svmbag
1000 samples
  16 predictors
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validation (10 fold)
Summary of sample sizes: 900, 900, 900, 900, 900, 900, ...

Resampling results

   Accuracy   Kappa   Accuracy SD   Kappa SD
    0.728      0.293    0.0444      0.132

Tuning parameter 'vars' was held constant at a value of 35
```

Kappa 值小于 0.3，看上去 bagging 的支持向量机模型的性能不如决策树模型。值得指出的是，Kappa 统计量的标准差（标记为 Kappa SD）比 bagging 决策树大很多。这说明性能随着不同的折会发生很大的改变。这种变化说明如果增加集成学习模型的数目，性能会有所提高。

11.2.3 boosting

另一种基于集成学习的受欢迎的方法是 **boosting**，因为它增加弱学习器的性能来获得强学习器的性能。这种方法主要基于 Rob Schapire 和 Yoav Freund 的大量工作，他们发表了很多关于这个主题的文章。



关于 boosting 的更多信息，请参考 *Boosting – Foundations and Algorithms* Understanding Rule Learners by R. Schapire, and Y. Freund, (The MIT Press, 2012)。

对于很多分类器，每一个的错误率都小于 50%。Schapire 和 Freund 发现，boosting 的结果经常相当好，至少不亚于它们当中最好的模型。实质上，它可以通过简单地添加更多弱学习器的方式来将性能提升到任意的阈值。因为这个发现有如此明显的作用，所以 boosting 方法被认为是机器学习领域最重要的发现之一。

类似于 bagging，boosting 也是使用在不同的重抽样数据中训练模型的集成学习，通过投票来决定最终的预测值。最关键的差异在于 boosting 中的重抽样数据集是专门构建用来产生

互补的模型，而且所有的选票并不是同等重要，会根据性能进行加权

boosting 的算法称为 AdaBoost 或者自适应 boosting，于 1997 年提出。该算法产生弱分类器来迭代地学习训练集中很大比例的难以分类的样本，对经常分错的样本进行更多的关注（也就是说给予更大的权重）。

从未加权的数据开始，第一个分类器尝试对结果建模。预测正确的样本出现在下一个分类器的训练集中的可能性比较小，相反地，难以分类的样本将会出现得更频繁。当下一轮的弱分类器被添加后，它们用来训练后面更难的数据。该过程会持续进行，直到达到要求的总误差或者性能不再提高。这时，每个分类器的票数会按照它们在建模数据集上的准确度进行加权。

虽然 boosting 的原理可以用于几乎任何模型，但是它在决策树中用得更多。我们已经在第 5 章使用过这种方式，用来提升 C5.0 决策树的性能。

AdaBoost.M1 算法提供了 AdaBoost 另一种基于树的实现。因为它与我们之前创建的 boosting 树比较类似，所以我们这里不介绍。



AdaBoost.M1 算法可以在 adabag 添加包中找到，更多信息可以参考：adabag – an R package for classification with boosting and bagging, Journal of Statistical Software, Vol 54(2), pp. 1-35, by E. Alfaro, M. Gamez, and N. Garcia (2013).

11.2.4 随机森林

另一种基于集成学习的方法称为随机森林（或者决策树森林），它只关注决策树的集成学习。该方法由 Leo Breiman 和 Adele Cutler 提出，将 bagging 和随机特征选择结合起来，对决策树模型添加额外的多样性。在树的集成（森林）产生之后，该模型使用投票的方法来组合预测结果。



关于如何构建随机森林的更详细的信息，请参考：Random forests, Machine Learning, Vol. 45, pp. 5-32, by L. Breiman (2001)。

随机森林将全能型和很强的能力结合到单一的机器学习方法中。因为集成学习只需要使用全体特征集中的一个很小的随机部分，所以随机森林可以处理非常大量的数据，而大数据中所谓的“维数灾难”常常会让其他的模型失败。与此同时，它对于大多数模型的误差率和任何其他方法处于同等的水平。



虽然随机森林的名称是 Breiman 和 Cutler 起的专用术语（详见 <http://www.stat.berkeley.edu/~breiman/RandomForests/>），但该名词有时候会用作任意类型的决策树集成的口语化表达。学会使用更通用的术语“决策树森林”，除非特指 Breiman 和 Cutler 的算法。

下表列出了随机森林模型优点和缺点。值得注意的是，相对于其他基于集成学习方法，

随机森林非常有竞争力而且在竞赛中具有核心优势。例如，随机森林更易于使用，并且具有更少的过拟合倾向。

优点	缺点
<ul style="list-style-type: none"> 对于大多数问题都很有有效的通用模型 	<ul style="list-style-type: none"> 与决策树不同，该模型不容易解释
<ul style="list-style-type: none"> 可以处理噪声和缺失值；分类和连续的特征 	<ul style="list-style-type: none"> 可能需要费工夫使得模型符合数据
<ul style="list-style-type: none"> 只选择最重要的特征 	
<ul style="list-style-type: none"> 可以适用于特征数目或者样本量极大的情况 	

由于它们的能力、多功能性和容易使用，随机森林很快成最受欢迎的机器学习算法之一。在本章的后面部分，我们将随机森林与 boosting C5.0 决策树进行一对一的比较。

1. 训练随机森林

虽然 R 中有好几个可以创建随机森林的添加包，但 `randomForest` 添加包可能是最忠实于 Breiman 和 Cutler 原创的添加包。还有一个额外的好处是得到了 `caret` 添加包中自动参数调整的支持。训练该模型的语法如下所示：

随机森林语法
使用 <code>randomForest</code> 添加包中的 <code>randomForest()</code> 函数
<p>创建分类器：</p> <pre>m <- randomForest(train, class, ntree = 500, mtry = sqrt(p))</pre> <ul style="list-style-type: none"> <code>train</code> 是包含训练数据集的数据框 <code>class</code> 是一个因子向量，代表训练集中每一行的类别 <code>ntree</code> 是一个整数，指定树的数目 <code>mtry</code> 是一个可选的整数，代表每次划分中随机选择的特征（变量）的数目（默认是 <code>sqrt(p)</code>，其中 <code>p</code> 是数据中总的变量数） <p>该函数将返回一个随机森林对象，可以用来进行预测。</p> <p>进行预测：</p> <pre>p <- predict(m, test, type = "response")</pre> <ul style="list-style-type: none"> <code>m</code> 是 <code>randomForest</code> 函数训练的模型 <code>test</code> 是包含测试集的数据框，与训练集数据的结构相同 <code>type</code> 可以是 <code>"response"</code>、<code>"prob"</code> 或者 <code>"votes"</code> 中的一个，分别表示输出的预测向量是否是预测类别、预测概率或者投票数的矩阵。 <p>该函数按照 <code>type</code> 参数的类型返回预测值。</p> <p>例子：</p> <pre>credit_model <- randomForest(credit_train, loan_default) credit_prediction <- predict(credit_model, credit_test)</pre>

之前提示过，默认情况下，`randomForest()` 函数创建一个 500 棵树的集合，每一个划分包含 `sqrt(p)` 个随机特征（`p` 是训练数据集中特征的个数）。这些参数是否合适取决于学习任务 and 训练数据的本质。一般来说，更复杂的问题和大量的数据（特征数和样本数都很大）使用大数量的树效果会更好。

使用大数量的树的目的是使得每一个特征都有机会在多个模型中被充分训练。它用

`mtry` 表示, 是 \sqrt{p} 默认值的基础。使用这个值可以对特征进行充分的限定, 从而使树与树之间发生大量的随机变化。例如, 因为信用数据 (credit data) 包含 16 个特征, 所以每棵树在任何时候被限定在 $\sqrt{16}=4$ 个特征上。

让我们看看默认的 `randomForest()` 参数如何作用于信用数据。我们将使用与其他学习器相同的方式来训练模型 (函数 `set.seed` 确保结果可重复)。

```
> library(randomForest)
> set.seed(300)
> rf <- randomForest(default ~ ., data = credit)
```

我们可以很简单地键入结果对象的名称来查看模型性能的汇总:

```
> rf

Call:
randomForest(formula = default ~ ., data = credit)
      Type of random forest: classification
      Number of trees: 500
No. of variables tried at each split: 4

      OOB estimate of error rate: 23.8%
Confusion matrix:
      no yes class.error
no   640   60  0.08571429
yes  178  122  0.59333333
```

正如期望的一样, 输出显示该随机森林包含 500 棵树, 对每个划分使用了 4 个变量。通过显示的混淆矩阵你可能会被看上去很差的重代入误差吓了一跳——23.8% 的错误率比起目前我们见过的任何集成学习方法都差。事实上, 这个混淆矩阵根本不是重代入误差。它其实代表出包错误率 (out-of-bag error rate, 标记为 OOB 估计错误率), 它是对测试集合错误的一个无偏估计。它表示对未来性能的一个合理估计。

对“出包” (out-of-bag) 估计是在构建随机森林时计算的。实质上, 任何没有选择的某棵树的自助法抽样中的样本都可以用来测试模型对未知数据的性能。在森林构建结束时, 每个样本每次的预测值会被记录, 通过投票来决定该样本最终的预测值。这种预测的总错误率就成了“出包”错误率。

2. 评估随机森林的性能

之前提到过, `randomForest()` 函数也得到了 `caret` 添加包的支持, 它允许在优化模型的同时计算“出包”错误率之外的性能度量指标。为了让事情更有趣, 我们比较自动调节参数的随机森林和之前做过的自动调节参数的 `boosting C5.0` 决策树模型。我们假设这个实验的目的是为了找出一个候选模型, 然后提交到一个机器学习的竞赛中。

首先加载 `caret` 添加包并设置训练控制选项。为了对模型性能进行最精确的比较, 使用重复 10 折交叉验证: 10 次 10 折交叉验证。这意味着模型会花费更多的时间和更大的计算量。因为这是最后一个比较, 所以我们非常肯定我们做了正确的选择——最后关头的胜利者

将会作为我们唯一的输入提交给机器学习竞赛。

```
> library(caret)
> ctrl <- trainControl(method = "repeatedcv",
                      number = 10, repeats = 10)
```

其次，我们对随机森林设置参数调整网格。该模型中唯一需要调整的参数是 `mtry`，它表示每一次划分中要随机选择多少个特征。我们知道默认是使用 $\sqrt{16}=4$ 个特征。为了更彻底，我们测试该数的一半、两倍以及所有的特征。因此，我们需要创建一个值为 2、4、8、16 的网格：

```
> grid_rf <- expand.grid(.mtry = c(2, 4, 8, 16))
```



当随机森林在每一次划分中用到所有特征时，实际上它与 bagging 决策树是一样的。

我们可以把结果的网格对象与 `ctrl` 对象一起传送给 `train()` 函数。我们使用 Kappa 值来选择最好的模型。

```
> set.seed(300)
> m_rf <- train(default ~ ., data = credit, method = "rf",
               metric = "Kappa", trControl = ctrl,
               tuneGrid = grid_rf)
```

之前的命令可能会花费很多时间，因为它要做很多工作！当它结束后，我们使用 10、20、30、40 次迭代来与 boosting 树进行比较：

```
> grid_c50 <- expand.grid(.model = "tree",
                        .trials = c(10, 20, 30, 40),
                        .winnow = "FALSE")
> set.seed(300)
> m_c50 <- train(default ~ ., data = credit, method = "C5.0",
               metric = "Kappa", trControl = ctrl,
               tuneGrid = grid_c50)
```

当 C5.0 决策树也计算完成后，我们可以一项一项地比较两种方法的差异。随机森林模型的结果如下：

```
> m_rf
```

Resampling results across tuning parameters:

mtry	Accuracy	Kappa	Accuracy SD	Kappa SD
2	0.725	0.128	0.0169	0.0636
4	0.75	0.293	0.0299	0.0877
8	0.754	0.338	0.0311	0.0835
16	0.756	0.361	0.0338	0.0889

Boosting C5.0 模型的结果如下：

```
> m_c50
```

Resampling results across tuning parameters:

trials	Accuracy	Kappa	Accuracy SD	Kappa SD
10	0.732	0.322	0.0402	0.0952
20	0.734	0.327	0.0403	0.0971
30	0.738	0.334	0.0367	0.0894
40	0.739	0.334	0.0393	0.0975

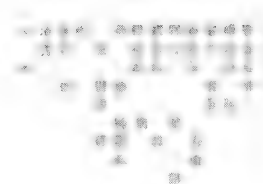
Kappa 值为 0.361, `mtry=16` 时的随机森林模型成了这 8 个模型中的赢家。它比最好的 C5.0 决策树模型要稍微好一些, Kappa 值是 0.334。基于这些结果, 我们将随机森林作为我们最终的模型提交上去。在没有通过竞赛数据进行真实评估的情况下, 我们没办法确定它最终是否会赢。但是根据我们的性能估计, 这是个安全的选择。如果足够走运的话, 我们可能会赢得奖金。

11.3 总结

在读完这一章后, 你现在已经知道了可以赢得数据挖掘或者机器学习竞赛奖金的基本技术。自动调节参数的方法可以帮助我们对单一模型尽可能地进行优化。另一方面, 创建一个机器学习模型的组, 让它们一起工作, 也能获得性能的提高。

虽然本章的目的是让你准备足够参加竞赛的模型, 但是要记住你的对手也能得到相同的技术。你不能停滞不前, 你需要不断努力, 在你的技能集上不断增添新的方法。你可以将领域的专门知识引入进来, 或者你的强项在于数据准备时对数据深入观察。任何情况下, 实践带来完美, 所以要尽可能地利用各种公开的竞赛来测试、评估和提高自己在机器学习方面的技能集。

在下一章中(本书的最后一章), 我们将使用 R 将机器学习方法应用到一些高度专业化和困难的领域进行鸟瞰。你将学到将机器学习应用到前沿领域的知识。



其他机器学习主题

现在，你可能迫不及待想要把机器学习应用到你自己的项目中去——你有可能已经这么做了。如果你已经在自己的项目上尝试过机器学习方法，你可能会发现，把数据变成实践远比本书所呈现得更难。

当你试图收集数据时，你可能意识到信息可能以专有电子表格格式或者以多个网络页面的格式存在。让事情变得更糟的是，在花费了几小时手工重新编排数据格式以后，计算机的内存可能已经用完，并且慢得动不了了。甚至 R 可能崩溃或者冻结你的机器。希望你不会被这些吓住。随着经验的增长，这些问题会慢慢变得容易处理。

本章涵盖了一些并不是所有机器学习项目中都会应用到的技术，但是可以证明这些技术对某些类型的工作还是有用的。你可能会发现，在你应用如下一些特征数据时，这些技术是特别有用的：

- 存储为无结构或者专有格式的数据，比如网页、网页 API 或者电子表格。
- 从生物信息学或者社交网络分析等这些有额外挑战领域得到的数据。
- 数据集大到 R 不能将它存储在内存中，或者机器学习需要花费很长的时间来完成。

如果你碰到这些问题，其实你并不是个例。尽管没有万能药（这些问题是数据科学家的难题，同时这也是对数据技术有大量需求的原因）但通过 R 社区的无私奉献，为解决这类问题的一系列 R 添加包提供了一种领先的方案。

本章提供了这类问题解决方案的一个简介。即使你是一个有经验的 R 语言老手，你也可能发现其中的某个添加包能简化你的工作流程，或许未来有一天你能开发一个简化所有人工作的添加包。

12.1 分析专用数据

与本书所分析的数据不同，真实世界的数据罕有打包为一个简单的能从网站上下载的 CSV 格式。真实的情况是，需要花费大量的工夫去分析准备数据。数据需要收集、合并、挑选、过滤或者重新格式化来满足学习算法的要求。这个过程俗称为**数据再加工** (data munging)。当特定的数据集从百兆字节增长到千兆字节，数据的来源是不相关且凌乱的，并且很多都是从特殊领域的来源收集而来时，再加工变得更加重要。下面的几节将介绍几个可以用在专用的或者特殊领域数据的添加包。

12.1.1 用 RCurl 添加包从网上获取数据

Duncan Temple Lang 开发的 RCurl 添加包为 curl 功能 (client for URL) 提供了一个 R 接口，一个在网络上传递数据的命令行工具。curl 功能在 Web 数据抓取上很有用，抓取是指从网站上收集数据并且把它们转变成结构化数据。



RCurl 添加包的资料可以在网站 <http://www.omegahat.org/RCurl/> 上获得。

在安装 RCurl 添加包以后，下载一个页面只需要输入一个简单的命令：

```
> library(RCurl)
> webpage <- getURL("http://www.packtpub.com/")
```

这将把 Packt 出版社主页上的全文 (包括所有网页的标签) 保存为一个名称为 webpage 的 R 字符对象。如下面命令行所示，尽管这样并不是很有用：

```
> str(webpage)
chr "<!DOCTYPE html>\n<html xmlns=\"http://www.w3.org/1999/xhtml\"
lang=\"en\" xml:lang=\"en\" >\n<head>\n  <title>Home | Packt Pu\" | _
truncated_"
```

因为大多数的网页应用特有的网页格式 (例如，XML/HTML 和 JSON 格式)，所以网页数据在应用到 R 中之前需要进行预处理。下面的几节中将讨论两个可以完成该任务的函数。

12.1.2 用 XML 添加包读 / 写 XML 格式数据

XML 是可读的纯文本文档，但是很多文档格式是基于结构化的通用标记语言。尤其是，很多基于网络的文档使用 XML 格式。XML 添加包是为读 / 写 XML 文档在流行的基于 C 的 libxml2 解析器基础上，提供了一组功能，它是由 Duncan Temple Lang 开发的。与 (前面提及的) RCurl 添加包结合在一起，就可能直接从网络上下载并且处理文档。



更多的关于 XML 添加包的信息，包括让你上手更快的简单实例，可以在网站：<http://www.omegahat.org/RXML/> 上找到。

12.1.3 用 rjson 添加包读 / 写 JSON

Alex Couture-Beil 开发的 `rjson` 添加包可以用来读 / 写 JSON (JavaScript Object Notation 格式的文件)。JSON 是一种标准的纯文本格式, 很多时候用作网页上的数据结构或对象。由于这种格式在创建网页应用上的实用性, 所以变得很流行。但是尽管其名字是这样的, 但它的作用并不限于网页浏览。



要想进一步了解 JSON 格式, 请访问: <http://www.json.org/>。

JSON 格式用纯文本字符存储对象。在安装了 `rjson` 添加包以后, 把 JSON 对象转换成 R 对象:

```
> library(rjson)
> r_object <- fromJSON(json_string)
```

要想把 R 对象转换成一个 JSON 对象:

```
> json_string <- toJSON(r_object)
```

使用 `RCurl` 添加包 (前面提及的), 可以用直接应用来源于很多在线数据库的 JSON 数据来编写 R 程序。

12.1.4 用 xlsx 添加包读 / 写 Microsoft Excel 电子表格

Adrian A. Dragulescu 的 `xlsx` 添加包提供了读 / 写 Excel 2007 (或者更早版本) 格式的电子表格的功能——这种电子表格广泛用于很多商业环境。这个添加包基于用来读 / 写 Microsoft 文档的 Apache POI Java API。



要想获得关于 `xlsx` 添加包的更多信息, 包括一个快速入门文档, 请参阅:

<https://code.google.com/p/rexcel/>。

12.1.5 生物信息学数据

由于基因数据具有独特的性质, 所以生物信息学领域的数据分析给出了不同于其他领域的一系列挑战。DNA 和蛋白质微阵列的使用导致数据集的宽度要大于其自身的长度 (即特征的数量超过案例数量)。当试图在这类数据上应用传统的可视化、统计检验和机器学习方法时, 就会产生问题。



基因统计或者生物信息学方面的 CRAN 任务视图可以在网址: <http://cran.r-project.org/web/views/Genetics.html> 获得。

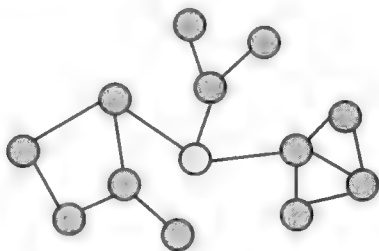
西雅图的 Fred Hutchinson 癌症研究中心的 **Bioconductor** 项目 (<http://www.bioconductor.org/>) 为分析基因数据提供了一个集中化的方法集合。该项目以 R 语言为基础, 添加了特别

针对生物信息学方面的添加包和文档。

Bioconductor 项目提供了分析来源于通用平台的微阵列数据的工作流程，例如来源于 Affymetrix、Illumina、Nimblegen 和 Agilent 等的微阵列数据。同时，它还提供了序列注释、多重检验法、专业的可视化和很多其他的功能。

12.1.6 社交网络数据和图数据

社交网络数据和图数据有很多挑战。这些数据记录了人或者对象之间的关系或者链接。如果有 N 个人，就有可能给出一个 $N \times N$ 的链接矩阵。随着人数的增加，它会变得非常复杂。用统计学方法和可视化方法来分析该网络，以便寻找有意义的关系模式。



由 Carter T. Butts、David Hunter 和 Mark S 开发的 network 添加包为处理这类网络提供了一种特殊的数据结构。一个密切相关的添加包 sna，也可以用来分析和可视化 network 对象。



要想了解更多关于 network 和 sna 的信息，请参考华盛顿大学的项目网站：
<http://www.statnet.org/>。

12.2 提高 R 语言的性能

R 语言有越变越慢和内存使用效率低下的名声，至少在某些方面这是名符其实的。现代计算机在处理几千个记录的数据集时，这种缺陷很大程度上是不被觉察的，但是 100 万或者更多记录的数据集则可能超出现在消费级别计算机硬件的承受极限。当数据有很多的属性或者用到了复杂的学习算法时，这个问题会变得更糟。



CRAN 有一个高性能计算任务视图，它列出了能够突破 R 语言所能完成极限的添加包：<http://cran.r-project.org/web/views/HighPerformanceComputing.html>。

扩展 R 语言基本添加包以使突破其基础包能力的添加包正在被迅速开发。这个工作主要从两方面来实现：有些添加包通过让数据运算得更快或者允许数据容量超过可用系统内存的大小，从而增加处理极大数据集的能力；其他的可能是通过应用特殊的计算机硬件，或者通过提供对大数据问题进行优化的机器学习算法，把工作分配到额外的计算机或者处理器中，

从而让 R 语言运行得更快。下面列出了其中的一些添加包。

12.2.1 处理非常大的数据集

非常大的数据集有时候会导致系统没有内存来存储，从而导致 R 语言慢慢终止。即使整个数据集能放在内存中，它也需要额外的内存从磁盘里读取数据，这就需要整个内存的大小要远大于数据集本身。另外，除了要处理非常大的记录外，非常大的数据会导致花费很长的时间来处理其他无缘无故的东西；即使在执行了几百万次操作时，也可能会附加一个短暂的运算。

几年前，很多人建议在 R 语言以外的其他程序语言中执行巨大数据集的数据准备，然后用 R 语言对一个相对较小的子集进行分析。然而，现在再也不需要这样做了，因为有几个添加包有助于 R 语言处理这些大数据问题。

1. 用 `data.table` 添加包使数据框运算得更快

由 Dowle、Short 和 Lianoglou 开发的 `data.table` 添加包提供了一种数据框的增强版本，叫做数据表（data table）。数据表在构造子集、连接和分组运算上普遍比数据框更快。然而，因为它本质上是一个改进的数据框，所以它的结果对象能够用到所有接受数据框的 R 函数中。



数据表项目的具体信息在下列网址中找到：<http://datatable.r-forge.r-project.org/>。

数据表结构的一个限制与数据框一样，它们受到了可用系统内存的限制。下面两小节讨论以牺牲与很多其他 R 函数兼容性为代价来战胜这个缺点的添加包。

2. 用 `ff` 添加包构建基于磁盘的数据框

由 Daniel Adler、Christian Glaser、Oleg Nenadic、Jens Oehlschlagel 和 Walter Zucchini 开发的 `ff` 添加包为数据框（`ffdf`）提供了另外的选择，它允许构建超过 20 亿行的数据集，即使这个远远超过了可用系统内存的大小。

`ffdf` 结构有一个物理部分（把数据以高效的形式存储在磁盘上）和一个虚拟部分（与一般的 R 数据框一样，但是明确指向存储在物理部分的数据）。你能够把 `ffdf` 对象想象成一张地图，指向磁盘上数据的位置。



`ff` 项目的网址：<http://ff.r-forge.r-project.org/>。

`ffdf` 数据结构的一个缺点是，它们不能被大多数的 R 函数应用。相反，数据要被处理成小块，稍后将结果结合在一起。给数据分块的优点是工作能分配给多个处理器同时使用并行计算方法处理，这个在接下来的几节中会提到。

由 Edwin de Jonge、Jan Wijffels 和 Jan van der Laan 开发的 `ffbase` 添加包处理这个问题与用 `ff` 对象提高基本的统计分析性能样。这让直接用 `ff` 对象探索数据变为可能。



`ffbase` 项目的网址为 <http://github.com/edwindj/ffbase>。

3. 用 `bigmemory` 添加包使用大矩阵

由 Michael J. Kane 和 John W. Emerson 开发的 `bigmemory` 添加包允许使用超过可用系统内存的极端大的矩阵。这个矩阵能存储在磁盘上或者共享存储器中，允许它们用在同一计算机上或者通过网络的其他进程中。这促进了并行计算方法，如下文提及的那样。



`bigmemory` 添加包的资料可以在 <http://www.bigmemory.org/> 上找到

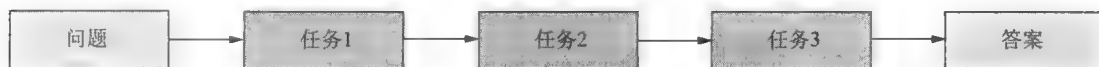
因为 `bigmemory` 矩阵与数据框不同，所以它们不能直接用到本书所提及的大多数机器学习算法中。它们也只能和数值型数据一起使用，也就是说，因为它们和典型的 R 矩阵类似，所以很容易构建能可以转化成标准的 R 数据结构的较小的案例或者小块。

上述作者同时也提供了 `bigalgebra`、`biganalytics` 和 `bigtabulate` 三个添加包，它们使在矩阵上执行简单的分析成为可能。特别注意的是，`biganalytics` 添加包中的 `bigkmeans()` 函数，它可以执行第 9 章所讨论的 k 均值聚类任务。

12.2.2 使用并行处理来加快学习过程

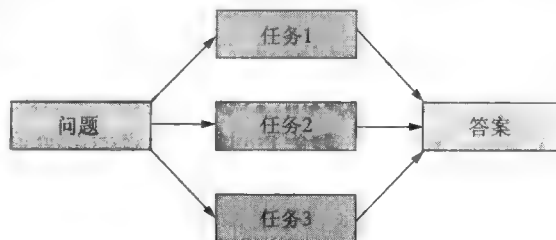
在早期的计算机处理中，程序完全串行的，这限制它们每一次只能执行一个任务。下一条指令只有在上一条指令完成的情况下才能执行。然而，很多任务可以通过让多个工作同时进行来更有效地完成。

串行计算：



这个需求在并行计算方法发展起来以后得到了解决，它用一组（两个或者更多）处理器或者计算机来解决一个更大的问题。很多现代计算机就是为并行计算而设计的。即使在只有一个处理器的情况下，它们也有两个或者更多的能并行工作的内核。这使得任务可以一个个独立地完成。

并行计算：



称为群集（cluster）的多个计算机的网络，也可用于并行计算。一个大的群集可

能包括各种硬件，并且硬件之间可以隔开很长的距离。在这种情况下，群集也能被称为**网格**（grid）。极端情况下，运行着硬件的标准成百上千个计算机的群集或者网格可能是一个非常强大的系统。

然而，重点是不是每一个问题都能被并行处理；有些问题相对来说更适合并行处理。你可能会想，加上了 100 个处理器后会使相同时间里完成的工作量有 100 倍的提升（即执行时间是 $1/100$ ），但是这显然是不可能的。因为要花费精力来管理工人：工作一开始必须要被分为不重叠的任务，然后将每个工人的结果结合成一个答案。

所谓的**易并行**（embarrassingly parallel）问题是理想化的。这些任务很容易缩减为各个非重叠的工作部分，并且结果很容易再次结合在一起。一个易并行机器学习的例子是 10 折交叉验证。一旦决定了样本，10 个估计中每一个都是独立的，这意味着每一个的结果不会影响其他的结果。正如你将看到的，使用并行计算后，任务运行速度会有显著的提高。

1. 度量运行时间

如果无法系统地度量节约了多少时间，那么加快 R 语言运行速度所花费的努力就会被浪费。尽管你也能坐在哪里、看着时钟计算时间，但是一种简单的解决方法是把惹人厌的代码都放在 `system.time()` 函数中。

例如，在作者的笔记本电脑中，`system.time()` 函数显示它花费了 1.3 秒来产生 100 万个随机数：

```
> system.time(rnorm(1000000))
      user  system elapsed 
    0.13    0.00    0.13
```

这个函数也可以用来评价性能改进了多少，它可以用我们已经提过的方法或者任何 R 函数来获得。

2. 用 **foreach** 添加包来处理并行问题

Revolution Analytics 公司的 Steve Weston 开发的 **foreach** 添加包提供了可能是最简单的启动并行计算的方法。尤其是，如果在 Windows 操作系统上运行 R 语言，因为有些添加包只针对特定的平台。

这个添加包的核心是一个新的 **foreach** 循环结构。如果你用过其他的编程语言，那你可能会很熟悉它。本质上，它允许在一组中的多项上循环，不用清楚地数出项的数量。换句话说，组中的每项都会用到。



除了 **foreach** 添加包外，Revolution Analytics 公司开发了高性能的、适合企业的 R 项目。免费版本是针对实验和学术用途。要想了解更多的信息，查看网站：<http://www.revolutionanalytics.com/>。

如果你认为 R 已经提供了一组在项上循环的应用函数（例如，`apply()`、`lapply()`）

和 `sapply()` 等), 那么你是正确的。但是, `foreach` 循环有另一个好处: 用非常简单的句法就能完成并行的循环迭代。

其姐妹添加包 `doParallel` 为 `foreach` 添加包提供了一个并行的后端, 它利用 R 语言 (2.14.0 版本及后续版本) 内置的 `parallel` 添加包。 `parallel` 添加包包括 `multicore` 添加包和 `snow` 添加包两部分, 这在下面的章节中描述。

3. 用 `multicore` 添加包实现一个多重任务操作系统

Simon Urbanek 的 `multicore` 添加包允许在一台有多个处理器或者处理内核的计算机上并行处理。因为它利用了操作系统的多重任务处理能力, 所以它并不是被 Windows 系统本身支持的。一种使用 `code` 添加包的简单方法是用 `mcapply()` 函数, 它是 `lapply()` 函数的一个并行版本。



`multicore` 项目发布在 <http://www.rforge.net/multicore/> 上。

4. 用 `snow` 和 `snowfall` 添加包来联网多个工作站

Luke Tierney、A. J. Rossini、Na Li 和 H. Sevcikova 的 `snow` 添加包允许在多核或多处理器的机器以及多台机器的网络上并行处理。Jochen Knaus 的 `snowfall` 添加包提供了一个比 `snow` 更好操作的接口。



要想了解更多关于代码的信息, 包括一个详细的 FAQ 和关于如何在网络上配置并行计算, 请参阅网址: <http://www.imbi.uni-freiburg.de/parallel/>。

5. 用 `MapReduce` 和 `Hadoop` 添加包进行并行云计算

`MapReduce` 编程模型是由 Google 发布的, 它用于处理在联网计算机的大集群上的数据。`MapReduce` 把并行编程定义为一个两步过程:

- ❑ `map` 步骤, 在这个步骤中, 将问题分成一个个相对较小的任务并分配给集群中的计算机。
- ❑ `reduce` 步骤, 在这个步骤中, 收集各个小块工作的结果并且合成为原始问题的最终解决方案。

替代专利 `MspReduce` 框架的一个流行的开源选择是 `Apache Hadoop`。`Hadoop` 软件由 `MapReduce` 概念和一个能在计算机集群之间存储大量数据的分布式文件系统组成。



Packt 出版社出版了多本关于 `Hadoop` 的书。要想预览关于这个主题的书的列表, 请参考下面的 URL: <http://www.packtpub.com/books?keys=Hadoop>。

多个提供 R 语言到 `Hadoop` 接口的 R 项目正在开发。其中一个 `Saptarshi Guha` 的

RHIPE 添加包，它解决 R 和 Hadoop 之间的通信试图把分配和再结合的思想带到 R 语言中。



RHIPE 添加包暂时不能在 CRAN 找到，只能从网络得到的源码来安装它，网址为 <http://www.datadr.org>。

Revolution Analytics 公司的 RHadoop 项目提供了一个到 Hadoop 的接口。这个项目提供了一个 rmr 添加包，目的是作为 R 开发者写 MapReduce 程序的简单途径。另外，RHadoop 添加包提供了 R 函数来访问 Hadoop 的分布式数据存储。

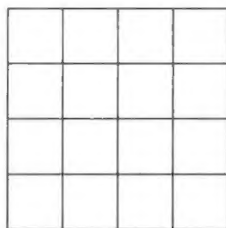


到本书出版时，RHadoop 的开发进行得非常迅速。要想了解更多这个项目的信息，请参见：<https://github.com/RevolutionAnalytics/RHadoop/wiki>。

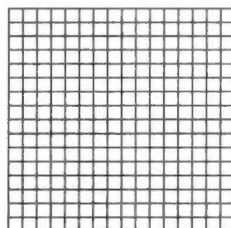
12.2.3 GPU 计算

并行处理的另一个选择是使用计算机图形处理单元（GPU）来增强数学计算速度。GPU 是在迅速把图像展现到计算机屏幕方面进行优化的专用处理器。因为计算机一直需要展现复杂的 3D 图像（尤其是为电子游戏），所以很多 GPU 用到了为并行处理设计的硬件和极其有效的矩阵和向量计算。一个附加的优点是它们能用来有效地解决某些类型的数学问题。计算机处理器可能是 16 核，而一个 GPU 可能有几千个核。

GPU 计算的缺点是它需要很多计算机都没有的专用硬件。在很多情况下，制造商 Nvidia 的 GPU 是必不可少的，因为他们提供了一个叫做 CUDA（Complete Unified Device Architectue）的专利框架，使得 GPU 可以使用普通的编程语言，比如 C++ 来编程。



有16个核的CPU



有1000个以上核的GPU



要想了解更多关于 Nvidia 在 GPU 计算中角色的信息，请访问 <http://www.nvidia.com/object/what-is-gpu-computing.html>。

由 Josh Buckner、Mark Seligman 和 Justin Wilson 开发的 gputools 添加包用 Nvidia CUDA 工具箱实现了多个 R 函数，比如，矩阵操作、聚类 and 回归模型。这个添加包要求 CUDA 1.3 或者更高版本的 GPU，并安装 Nvidia CUDA 工具箱。

12.2.4 部署最优的学习算法

本书所提到的一些机器学习算法能够运用在极其大的数据集上而仅需要相对较小的修

改。例如，用前面所提及的一个大数据添加包可以直接实现朴素贝叶斯或者 Apriori 算法。有些类型的模型，比如集成学习，本身就很适合于并行化，因为每个模型的工作可以分配给集群中的处理器或者计算机。另一方面，其他一些算法还是要求对数据或者算法进行很大的改变，或者需要在能用到巨大的数据集前，完全重新考虑。

本节将观察我们学习至今提供的最优的学习算法版本的添加包。

1. 用 **biglm** 添加包建立更大的回归模型

由 Thomas Lumley 开发的 **biglm** 添加包为不能适应内存的大数据集上训练回归模型提供了函数。它通过迭代过程，用小块的数据让模型一点点地更新。计算的结果将近乎与在整个数据集上运行传统的 `lm()` 函数得到的结果相似。

`biglm()` 函数允许用 SQL 数据库来代替数据框。模型也能用从前面提到的 `ff` 添加包中创建的数据对象中得到的小块来训练。

2. 用 **bigrf** 添加包建立更大、更快的随机森林模型

由 Aloysius Lim 开发的 **bigrf** 添加包实现了在特大数据集上进行分类和回归任务的随机森林，该类数据集太大因而无法应用本章前面提到的 **bigmemory** 对象把该数据集置于内存中。这个添加包也可以用前面提及的 **foreach** 添加包来进行快速并行处理。树可以并行地开枝散叶（在一台计算机或者多台计算机上）。就像真正的森林一样，额外的树能随时被加到森林中或者与其他的森林合并。



要想了解更多的信息，包括例子和 Windows 安装指南，请参见 GitHub 上的 wiki 添加包：<https://github.com/alloysius-lim/bigrf>。

3. 用 **caret** 添加包训练和评价模型

如果在 R 中注册过一个并行后端，由 Max Kuhn 开发的 **caret** 添加包（在第 10 章和第 11 章中广泛地提及）将透明地利用该并行后端（例如，用前面提及的 **foreach** 添加包）。

训练和评价模型涉及的许多任务（比如创建随机例子和为 10 折交叉验证重复地检验预测）都是高度平行的。这将建立一个特别好的 **caret**。



关于使用 **caret** 添加包进行并行处理、性能提升的配置说明和一个学习案例，可以在如下项目网址中得到：<http://caret.r-forge.r-project.org/parallel.html>。

12.3 总结

研究机器学习无疑是令人兴奋的。在并行和分布式计算这一相对前沿方向上的工作，给

出了探索大数据中未知知识的巨大潜能。迅速增长的数据科学社区被免费和开源的 R 语言所促进，这为人们接触 R 语言提供了一个很低的门槛——你只要愿意学习。

本书所介绍的内容为你理解更多高级机器学习方法奠定了基础。现在是你的任务，继续学习并且为你的知识工厂增加供给。沿途要记得“天下没有免费午餐”理论——没有一个机器学习模型是可以适用于所有数据集的。机器学习上总有人为的因素，我们要增加学科知识，具备使合适的算法和手中的任务相匹配的能力。

未来，随着机器学习和人类学习的界限逐渐模糊，将会很有趣地看到人类是如何改变的。像亚马逊 Mechanical Turk 这样的众包服务，它们基于人力市场中的智能，提供了短时间内完成简单任务的人类智力集群。也许有一天，当我们已经用计算机来执行人类不能简单完成的任务时，计算机会让人类做相反的事：深思。